AD-A040 238 IBM CORP CAPE CANAVERAL FLA F/G 5/9
DESIGN OF TRAINING SYSTEMS. TRAPAC USER'S MANUAL, DATA BASE, SC--ETC(U)
SEP 76 K V BRANCH, J D STALEY, R E YANKO N61339-73-C-0097
UNCLASSIFIED TAEG-36 NL

1 OF 6
AD A040238



AD A040238

TAEG

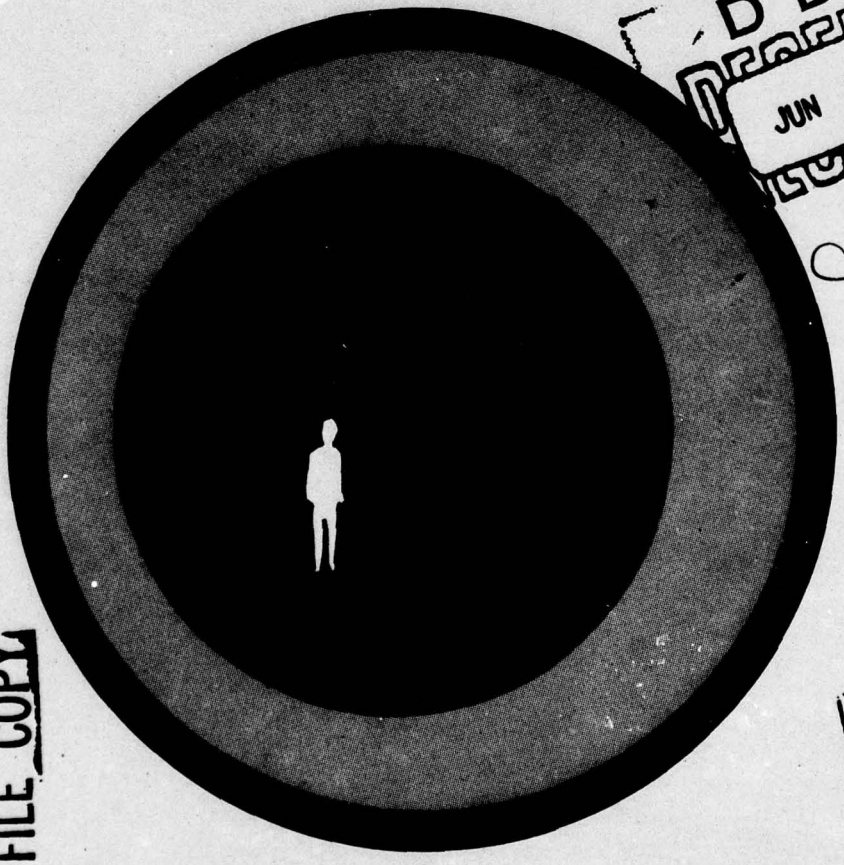
TRAINING ANALYSIS AND EVALUATION GROUP

TAEG REPORT NO. 36

DESIGN OF TRAINING SYSTEMS TRAPAC USER'S MANUAL

DATA BASE, SCRR, AND TPF MODELS

DDC
JUN 3 1977
C



FOCUS ON THE TRAINED MAN

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

DDC FILE COPY

SEPTEMBER 1976

TRAINING ANALYSIS AND EVALUATION GROUP
ORLANDO, FLORIDA 32813

Technical Report: TAEG REPORT NO. 36

DESIGN OF TRAINING SYSTEMS

TRAPAC USER'S MANUAL

ABSTRACT

This user's manual contains both a user-oriented guide to Design of Training Systems (DOTS) software use and application and programmer-oriented DOTS software descriptions. The major DOTS subsystems addressed include the DOTS data base and maintenance system, the System Capabilities/Requirements and Resources (SCRR) Model, and the Training Process Flow (TPF) model.

The user-oriented sections contain discussions of subsystem architecture, design assumptions, input requirements and output parameters. Procedures are provided for pre-application system test and for operational use. Both administrative and operational maintenance procedures are presented in the DOTS data base section.

The programmer-oriented sections present program descriptions and macro flows, input/output data formats (where applicable), and program source code listings for each subsystem (the two models and the data base). The information contained in these sections is intended for use by programmers tasked with installing, modifying, or maintaining the DOTS software.

The programs described in this manual were developed by IBM Corporation for the Training Analysis and Evaluation Group, Orlando, Florida (Contract No. N61339-73-C-0097).

Reproduction of this publication in whole or in part is permitted for any purpose of the United States Government.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER (18) TAEG Report No. (19) 36 — 2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) Design of Training Systems. TRAPAC User's Manual, DATA BASE, SCRR, and TPF Models,	5. TYPE OF REPORT & PERIOD COVERED
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) Kenneth V. Branch, James D. Staley Ronald E. Yanko	8. CONTRACT OR GRANT NUMBER(s) (15) N61339-73-C-0097
9. PERFORMING ORGANIZATION NAME AND ADDRESS International Business Machines IBM Corp. 7900 N. Astronaut Boulevard Cape Canaveral, Florida 32920	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Training Equipment Center Training Analyis and Evaluation Group Orlando, Florida	12. REPORT DATE (11) Sept 1976 — 13. NUMBER OF PAGES ~~578~~ (12) 576 p.
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report) UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release;
Distribution is Unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

DDC RECEIVED JUN 3 1977 C

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Training Management Systems
Computer Based Models
Training Resources Evaluation Modeling
Educational Technology
Training Technology
Interactive Training Data Base

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This user's manual contains both a user-oriented guide to Design of Training Systems (DOTS) software use and application and programmer-oriented DOTS software descriptions. The major DOTS subsystems addressed include the DOTS data base and maintenance system, the System Capabilities/Requirements and Resources (SCRR) Model, and the Training Process Flow (TPF) model. → next page mls

New

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

410210

20. ABSTRACT (Cont'd)

cont

The user-oriented sections contain discussions of subsystem architecture, design assumptions, input requirements and output parameters. Procedures are provided for pre-application system test and for operational use. Both administrative and operational maintenance procedures are presented in the DOTS data base section.

The programmer-oriented sections present program descriptions and macro flows, input/output data formats (where applicable), and program source code listings for each subsystem (the two models and the data base). The information contained in these sections is intended for use by programmers tasked with installing, modifying, or maintaining the DOTS software.

TAEG REPORT NO. 36

DESIGN OF TRAINING SYSTEMS

TRAPAC USER'S MANUAL

DATA BASE, SCRR, AND TPF MODELS

Kenneth V. Branch
James D. Staley
Ronald E. Yanko

This Report Was Prepared by

International Business Machines Corporation

for the

Training Analysis and Evaluation Group

September 1976

ACCESSION for
NTIS White Section
DDC Buff Section
UNANNOUNCED
JUSTIFICATION
BY
DISTRIBUTION/AVAILABILITY CODES
Dist. AVAIL. and/or SPECIAL
A 23

TABLE OF CONTENTS

SECTION		PAGE
I	INTRODUCTION	
	PURPOSE	I-1
	ORGANIZATION	I-1
	SYSTEM APPLICATION/OVERVIEW	I-2
	GENERAL SYSTEM PROCEDURES	I-2
	EQUIPMENT ENVIRONMENT	I-4
	PROGRAM ENVIRONMENT	I-5
II	DOTS DATA BASE USER GUIDE	
	INTRODUCTION	II-1
	DATA BASE ARCHITECTURE	II-1
	DATA BASE STRUCTURE	II-3
	DATA BASE DATA FIELD CONTENTS	II-7
	GENERAL MAINTENANCE PROCEDURES	II-15
	DATA BASE MAINTENANCE OPERATIONAL PROCEDURES - DATA ENTRY	II-18
	COURSE FILE MAINTENANCE	II-20
	INSTRUCTOR FILE MAINTENANCE	II-43
	FACILITY FILE MAINTENANCE	II-54
	DATA BASE MAINTENANCE OPERATIONAL PROCEDURES - CASSETTE TRANSMISSION TO NCSS	II-62
	TRANSFER DATA FROM TAPE TO DISK	II-62
	SORT "DB-INPUT"	II-62

PRECEDING PAGE BLANK-NOT FILMED

TABLE OF CONTENTS

SECTION		PAGE
	LIST SORTED TRANSACTION FILE	II-63
	TRANSMIT TRANSACTION FILE TO NCSS ENVIRONMENT	II-63
	DATA BASE MAINTENANCE OPERATIONAL PROCEDURES - PROCESSING AT THE NCSS COMPUTER	II-67
	DATA BASE MAINTENANCE OPERATIONAL PROCEDURES - GENERAL PURPOSE UTILITY PROGRAMS	II-71
III	TRAINING PROCESS FLOW MODEL - MODEL DESCRIPTION AND USER GUIDE	
	TPF MODEL ARCHITECTURE	III-1
	MODEL DESCRIPTION	III-2
	SUPPORT PROGRAMS	III-4
	INPUT PARAMETERS DESCRIPTION	III-7
	OUTPUT PARAMETERS DESCRIPTION	III-10
	TPF MODEL OPERATIONAL PROCEDURES	III-13
	GENERAL SYSTEM FLOW	III-13
	OPERATIONAL EXAMPLE	III-21
	TPF OPERATIONAL TEST PROCEDURES	III-24
IV	SYSTEM CAPABILITIES/REQUIREMENTS AND RESOURCES MODEL - MODEL DESCRIPTION AND USER GUIDE	
	SCRR MODEL ARCHITECTURE	IV-1
	MODEL DESCRIPTION	IV-2
	MODEL ASSUMPTIONS	IV-11

TABLE OF CONTENTS

SECTION		PAGE
	INPUT PARAMETERS DESCRIPTION	IV-15
	OUTPUT PARAMETERS DESCRIPTION	IV-18
	INSTRUCTOR ASSIGNMENT LISTING	IV-19
	FACILITY UTILIZATION LISTING	IV-19
	LP OPTIMAL SOLUTION AND SENSITIVITY ANALYSIS - RESOURCE DATA	IV-19
	LP OPTIMAL SOLUTION AND SENSITIVITY ANALYSIS - COURSE DATA	IV-27
	SCRR MODEL OPERATIONAL PROCEDURES	IV-30
	SELECT ORGANIZATIONAL ELEMENT	IV-32
	SCRR MODEL EXECUTION	IV-32
	INSTRUCTOR ASSIGNMENT LISTING GENERATION	IV-35
	FACILITY UTILIZATION LISTING GENERATION	IV-36
	RESOURCE AVAILABILITY MODIFICATION.	IV-37
	CONSTANT COURSE CONVENINGS/OPTIMAL COURSE CONVENINGS MODIFICATION	IV-38
	COURSE PARAMETER MODIFICATION	IV-39
	ADDITIONAL DATA MODIFICATION	IV-40
	INSTRUCTOR REQUIREMENTS COMPUTATION	IV-43
	SCRR MODEL OPERATIONAL TEST PROCEDURES	IV-46
	SCRR TEST DATA BASE DESCRIPTION	IV-46
	OPERATIONAL TEST PROCEDURES	IV-50
V	DOTS DATA BASE PROGRAM DESCRIPTIONS	
	INTRODUCTION	V-1
	GENERAL SYSTEM PROCEDURES.	V-1

TABLE OF CONTENTS

SECTION		PAGE
	RAMIS DATA MANAGEMENT SYSTEM FILE DESCRIPTIONS .	V-3
	TYPES OF MAINTENANCE ACTIVITIES	V-19
	MAINTENANCE DATA ENTRY PROGRAM	V-23
	MAINTENANCE TAPE CASSETTE LISTING PROGRAM . .	V-65
	MAINTENANCE DATA FILE TRANSMISSION PROGRAM . .	V-75
	NCSS MAINTENANCE TRANSACTION PROCESSOR . . .	V-85
	NCSS SOFTWARE NAMING CONVENTIONS	V-88
	RAMIS INDICATORS IN RUN PROCEDURE MESSAGES .	V-88
	NCSS MAINTENANCE TRANSACTION PROCESSING PROGRAMS	V-90
	NCSS PLI PROGRAMS	V-171
	GENERAL PURPOSE DATA BASE EXECUTIVES. . . .	V-206
VI	TRAINING PROCESS FLOW MODEL - PROGRAM DESCRIPTIONS	
	MODEL DESCRIPTION	VI-1
	SUPPORT PROGRAMS	VI-1
	DATA BASE SUPPORT PROGRAMS	VI-3
	"WHAT IF" SUPPORT PROGRAMS	VI-11
	COURSE MODIFICATIONS	VI-11
	ACROSS THE BOARD MODIFICATIONS	VI-15
	DISPLAY SUPPORT PROGRAM	VI-35
	EXECUTIVE DISPLAY	VI-35
	PROGRAM DISPLAY	VI-35
	TPF LOGIC DESIGN	VI-40

TABLE OF CONTENTS

SECTION		PAGE
	PROGRAM LOGIC DESIGN	VI-40
	TPF MODEL MODEL MAIN	VI-40
	SUBROUTINE CRSLS	VI-40
	SUBROUTINE PRERUN	VI-43
	SUBROUTINE CRSEND	VI-43
	SUBROUTINE QTREND	VI-43
	SUBROUTINE YEARND	VI-44
	SUBROUTINE CRTOUT	VI-44
	SUBROUTINE NPRINT	VI-44
VII	SYSTEM CAPABILITIES/REQUIREMENTS AND RESOURCES MODEL - PROGRAM DESCRIPTIONS	
	MODEL DESCRIPTION	VII-1
	SETSCH EXEC DESCRIPTION	VII-6
	SETSCH FORTRAN DESCRIPTION	VII-6
	SCRR EXEC DESCRIPTION	VII-9
	RAMSCRR EXEC DESCRIPTION	VII-12
	SCRR INST1 and INST2 DESCRIPTIONS . . .	VII-12
	SCRR FACIL DESCRIPTION	VII-13
	SCRR FAVAIL DESCRIPTION	VII-13
	SCRR IAVAIL DESCRIPTION	VII-13
	SCRR CRSE DESCRIPTION	VII-14
	BLANK EXEC DESCRIPTION	VII-20
	PGM94B PLI DESCRIPTION	VII-20
	PGM94A PLI DESCRIPTION	VII-20

TABLE OF CONTENTS

SECTION	PAGE
INTERFACE (IF1, IF2, IF3, AND IF4) EXECS OVERVIEW	VII-24
IF1 EXEC DESCRIPTION	VII-26
IF1 FORTRAN DESCRIPTION	VII-26
IF2 EXEC DESCRIPTION	VII-28
IF2 FORTRAN DESCRIPTION	VII-28
SCRRSORT EXEC DESCRIPTION	VII-32
IF3 EXEC DESCRIPTION	VII-33
IF3 FORTRAN DESCRIPTION	VII-33
IF4 EXEC DESCRIPTION	VII-35
IF4 FORTRAN DESCRIPTION	VII-35
COMB EXEC DESCRIPTION	VII-37
SCRR1 EXEC DESCRIPTION	VII-38
SCRR RUN DESCRIPTION	VII-38
SCRRFORM FORTRAN DESCRIPTION	VII-41
NCONT FORTRAN DESCRIPTION	VII-42
PGM93A PLI DESCRIPTION	VII-42
ASSIGN EXEC DESCRIPTION	VII-53
ASSIGN1 FORTRAN DESCRIPTION	VII-53
ASSIGN2 FORTRAN DESCRIPTION	VII-53
FACILITY EXEC DESCRIPTION	VII-59
FAC1 FORTRAN DESCRIPTION	VII-59
FAC2 FORTRAN DESCRIPTION	VII-60
MODAVAIL EXEC DESCRIPTION	VII-65
MODIFY FORTRAN DESCRIPTION	VII-65

TABLE OF CONTENTS

SECTION		PAGE
	REQMT EXEC DESCRIPTION	VII-69
	REQMT RUNEXEC DESCRIPTION	VII-69
	REQMT1 RUNEXEC DESCRIPTION	VII-69
	REQMT2 RUNEXEC DESCRIPTION	VII-70
	SCRR2 EXEC DESCRIPTION	VII-76
	FIXCONV EXEC DESCRIPTION	VII-77
	MAXCONV EXEC DESCRIPTION	VII-89
	SCRR3 EXEC DESCRIPTION	VII-79
	ICOMP EXEC DESCRIPTION	VII-80
	ICOMP RUNEXEC DESCRIPTION	VII-80
APPENDIX A	ALPHABETIZED PROGRAM LISTING INDEX	A-1

TAEG REPORT NO. 36

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE
I-1	DOTS System Diagram	I-3
II-1	Data Base Maintenance Procedure	II-17
III-1	TPF Model - Simplified System Flow	III-3
III-2	TPF Model - Functional Flow.	III-5
III-3	Sample Course, School, and Center Reports	III-11
III-4	TPF Model Baseline Run	III-22
III-5	Sample Problem: Reduce No Shows and Course Length	III-25
III-6	Sample Probelm: Reduce No Shows and Course Convenings	III-26
III-7	TPF Test Run Procedure	III-28
III-8	TPF Test Run - Course Output	III-28
III-9	TPF Test Run - School and Center Output	III-29
IV-1	SCRR Model Functional Flow	IV-3
IV-2	Instructor Assignment Listing	IV-20
IV-3	Facility Utilization Listing	IV-21
IV-4	LP Optimal Solution and Sensitivity Analysis - Resource Data	IV-23
IV-5	LP Optimal Solution and Sensitivity Analysis - Course Data	IV-28
IV-6	SCRR Model Operation Overview	IV-31
V-1	DOTS Facilities File Hierarchic Structure	V-3
V-2	DOTS Instructor File Hierarchic Structure	V-6
V-3	DOTS Course File Hierarchic Structure	V-11
V-4	RAMIS Maintenance Operations	V-87

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE
VI-1	TPF Model Simplified System Flow	VI-2
VI-2	TPF Model Data Base Interface	VI-4
VI-3	TPF Input Formats	VI-6
VI-4	TPF Model "WHAT IF" and Output Display	VI-12
VI-5	Flow Executive TPFMOD	VI-17
VI-6	Flow Executive TPF	VI-30
VI-7	Course Scheduler Matrix	VI-45
VI-8	Sample TPF Outputs	VI-46
VII-1	Overview of SCRR Executive Programs	VII-2
VII-2	SCRR EXEC Flowchart	VII-10
VII-3	RAMSCRR EXEC Flowchart	VII-15
VII-4	BLANK EXEC Flowchart	VII-21
VII-5	Data Base INTERFACE EXEC Flowchart	VII-25
VII-6	SCRR EXEC Flowchart	VII-43
VII-7	ASSIGN EXEC Flowcnart	VII-53
VII-8	FACILITY EXEC Flowchart	VII-61
VII-9	MODAVAIL EXEC Flowchart	VII-66
VII-10	REQMT EXEC Flowchart	VII-71

LIST OF TABLES

TABLE NO.	TITLE	PAGE
II-1	DOTS Facilities File Data Fields	II-7
II-2	DOTS Instructor File Data Fields	II-8
II-3	DOTS Course File Data Fields	II-10
II-4	Data Base Maintenance Transactions & Maintenance Transaction Files	II-68
II-5	Utility Maintenance File RAMIS Processing	II-70
III-1	RAMIS Dump Format (Master or Scratch)	III-8
V-1	DOTS Data Base Maintenance Transaction Types	II-20
VI-1	RAMIS Dump Format (Master of Scratch)	VI-5
VI-2	Listing TPFCOPY EXEC	VI-7
VI-3	Listing TPFSORT EXEC	VI-8
VI-4	Listing MPGM3 PLI	VI-9
VI-5	Listing PGMBO PLI	VI-10
VI-6	Listing Executive TPFMOD	VI-19
VI-7	Listing Program TPFMOD	VI-20
VI-8	Listing Subroutine DECODE	VI-22
VI-9	Listing Subroutine CONVRT	VI-26
VI-10	Listing Subroutine MSG	VI-27
VI-11	Listing Subroutine SHOW	VI-28
VI-12	Subroutine COMMD Listing	VI-29
VI-13	Listing Executive TPF	VI-31
VI-14	Listing Program PARAM	VI-32
VI-15	Listing Display Executives	VI-37

LIST OF TABLES

TABLE NO.	TITLE	PAGE
VI-16	Listing Program DISPLAY	VI-38
VI-17	Listing Program KEND	VI-47
VII-1	SCRR Executive Programs Description	VII-3
VII-2	SCRR Data File Summary	VII-5
VII-3	SCRR Program Listing	VII-11
VII-4	RAMSCRR Program Listings	VII-16
VII-5	Blank Program Listings	VII-22
VII-6	IF1 Program Listings	VII-27
VII-7	IF2 Program Listings	VII-30
VII-8	SCRRSORT Program Listing	VII-32
VII-9	IF3 Program Listings	VII-34
VII-10	IF4 Program Listings	VII-36
VII-11	COMB Program Listing	VII-37
VII-12	SCRR1 Program Listings	VII-44
VII-13	ASSIGN Program Listings	VII-56
VII-14	FACILITY Program Listings	VII-62
VII-15	MODAVAIL Program LIstings	VII-67
VII-16	REQMT Program Listings	VII-72
VII-17	SCRR2 Program Listings	VII-76
VII-18	FIXCONV Program Listing	VII-77
VII-19	MAXCONV Program Listing	VII-78
VII-20	SCRR3 Program Listing	VII-79
VII-21	ICOMP Program Listings	VII-81

SECTION I

INTRODUCTION

PURPOSE

This manual provides two levels of information on the Design of Training Systems (DOTS) models and the DOTS data base. The purpose of the first portion of the manual is to familiarize the non-ADP user of the DOTS software with the operation of the models and data base. Operational aspects of the models and data base are developed without detailed treatment of the design rationale or specific programming techniques employed in their implementation. The second half of the manual outlines explicit programming details. The level of detail presented in these sections will provide a technical base for programmers intending to install or modify the DOTS software.

ORGANIZATION

The TRAPAC User's Manual is organized into seven major sections covering operational and programming aspects of the DOTS data base, the Training Process Flow (TPF) model, and the System Capabilities Requirements and Resources (SCRR) model. The two models function independently and the data base (although it serves both the SCRR and the TPF models) can also be considered as an independent subsystem. Therefore, coverage of each of the models and the data base is found in separate sections.

Section II, the DOTS data base user guide, covers the structure of the data base including the data hierarchy employed in its construction. Operational procedures are divided into three types: the procedures necessary to record the maintenance data on a WANG data cassette at the activity level, procedures for transmitting the maintenance data via telecommunications to the timesharing environment, and procedures for processing the maintenance data in the timesharing environment.

Sections III and IV, which deal with the operational aspects of the models, are all organized similarly and include subsections on model architecture, operational procedures and operational test. The model architecture section describes the model for the user in terms of model functions, significant design assumptions, input data requirements and output data descriptions. Operational procedures are covered as they relate specifically to the individual models and contain the steps necessary to apply the models to appropriate problems. Operational tests are specified to the extent necessary to assess system functional integrity but do not cover tests to determine the validity of the system in specific applications.

Sections V, VI, and VII provide the detailed programming aspects of the DOTS software system. The data base is covered in Section V, the TPF model in Section VI, and the SCRR model in Section VII. The descriptions within

these sections encompass a textual and pictorial or graphical discussion accompanied by detailed program listings of the subsystem components.

SYSTEM APPLICATION/OVERVIEW

The DOTS system consists of two models supported by a three-file data base. Each model is of a different type and addresses a distinct facet in the development and management of training courses and systems. Figure I-1 is a functional level diagram of the DOTS system showing the interaction of the models and the data base.

The DOTS system is an interactive system. Each of the two models and the data base can be updated and used via terminal equipments in communication with a timesharing computer system. A brief description of the models is presented here. Detailed information on the models is contained in the body of this manual in Sections III, IV, VI, and VII. The three major parts of the system are:

a) Training Process Flow (TPF) model. The TPF model is an aggregate flow model designed to project the effect, on training system output, of changes in demand, scheduling, course capacity, and utilization.

b) System Capabilities Requirements and Resources (SCRR) model. A linear programming model designed to determine the maximum student throughput for a training complex, based on an optimal mix of course convenings and available resources.

c) The DOTS data base provides a common, direct access data source for both the SCRR and TPF models. The DOTS data base was constructed and is accessed via the RAMIS[1] system, a data base language developed by Mathematica, Inc.

GENERAL SYSTEM PROCEDURES

Detailed procedures for model and data base operations are contained in Sections II, III, and IV of this manual. General procedures are those which deal directly with the timesharing system. Procedures such as login, typing conventions and system level error recovery are included in the category of general procedures. It is assumed that the user will familiarize himself with the basic procedures for using the timesharing system prior to any attempt to use the DOTS system itself.

General system procedures are covered in the documentation provided as a part of the support supplied by National CSS. The user should be acquainted with the following National CSS documents:

[1] Trademark of Mathematica, Inc.

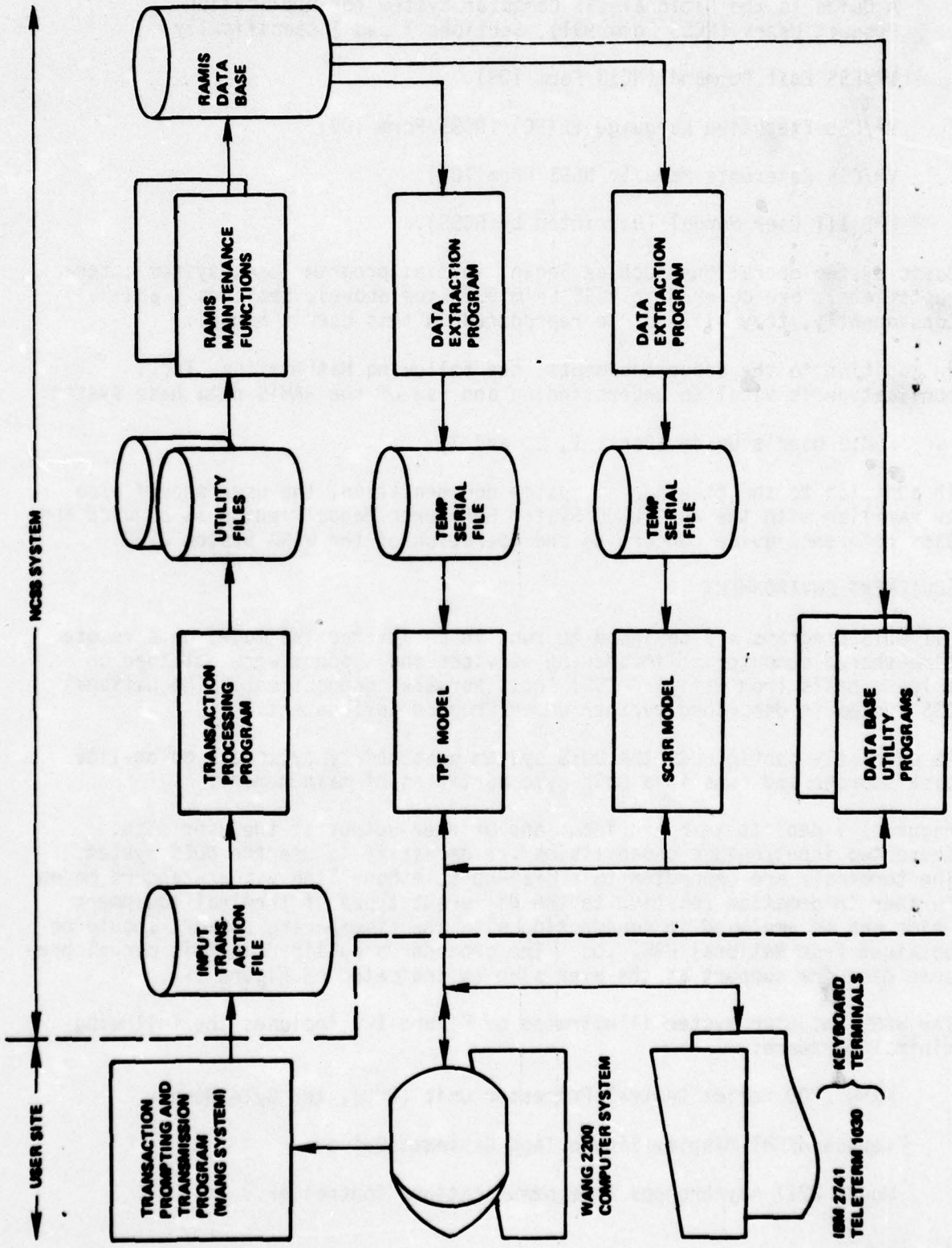


FIGURE I-1 DOTS SYSTEM DIAGRAM

A Guide to the National CSS Computer System for Application Product Users (NCSS Form 991), Sections 1 and 7 specifically.

VP/CSS Edit Command (NCSS Form 108).

VP/CSS Executive Language (EXEC) (NCSS Form 109).

VP/CSS Reference Manual (NCSS Form 106).

MPS III User Manual (Reprinted by NCSS).

Basic system operations such as login, initial program load, system interrupts, etc., are covered in NCSS Form 991 (see above), Sections 1 and 7. Consequently, they will not be reproduced in this User's Manual.

In addition to the above documents, the following Mathematica, Inc., publication is vital to understanding and use of the RAMIS data base system:

RAMIS User's Guide (Parts 1, 2, and 3).

In addition to the timesharing system documentation, the user should also be familiar with the WANG 2200 System Reference Manual, which is a quick and easy reference guide concerning the operation of the WANG System 2200.

EQUIPMENT ENVIRONMENT

All DOTS programs are designed to run, in an interactive mode, on a remote time-shared computer. Timesharing services and support were obtained on a lease basis from National CSS, Inc., Norwalk, Connecticut. The National CSS system is described further under Program Environment.

As currently configured, the DOTS system uses thirty cylinders on on-line disk storage and runs in a 320K byte partition of main memory.

Figure I-1 depicts keyboard input and printer output at the user site. These two input/output capabilities are necessary to use the DOTS system. The terminals are connected to a dial-up telephone line via a standard modem. Further information relative to the different types of terminal equipment which can be employed in conjunction with the timesharing support should be obtained from National CSS, Inc. The procedures outlined in this manual presume hardware support at the user site as indicated on Figure I-1.

The WANG computer system illustrated on Figure I-1 includes the following minimal hardware:

WANG 2200 Series Central Processor Unit (CPU), 16K Byte Memory.

Keyboard/CRT Display/Single Tape Cassette Drive.

Model 2227 Asychronous Telecommunications Controller.

The WANG computer system is required to make a data base transaction tape. The transmission of such a tape from the user site to the NCSS computer input transaction file also requires the WANG system. The WANG system has been programmed to perform as a passive keyboard ASCII terminal and can be used in this role in lieu of the other keyboard terminals shown on Figure I-1.

The IBM 2741 and TELETERM 1030[2] terminals shown on Figure I-1 can be used to initiate and control all DOTS procedures except those explicitly concerned with the makeup and transmission of the input transaction tape.

PROGRAM ENVIRONMENT

In addition to the National CSS timesharing software, the DOTS system employs several other program packages, all of which run under control of the VP/CSS[3] system. The program packages are:

Mathematical Programming System III: MPS III[4]

Rapid Access Management Information System: RAMIS[5]

[2]Trademark of Computer Devices, Inc.

[3]Trademark of National CSS, Inc.

[4]Trademark of Management Science Systems, Inc.

[5]Trademark of Mathematica, Inc.

SECTION II

DOTS DATA BASE

USER'S GUIDE

INTRODUCTION

The DOTS data base user's guide is organized into the following seven subsections:

a. Data base architecture

b. Data base structure

c. Data base data field contents

d. Data base maintenance procedures

e. Maintenance data entry procedures

f. Procedures for transmission of WANG data cassette to NCSS environment

g. Procedures for processing change data files at NCSS level

The first four subsections contain general background information on the DOTS data base. The last three subsections present detailed operating procedures for performing data base maintenance.

DATA BASE ARCHITECTURE

The DOTS data base consists of three independent data bases:

- o A Facilities data base (FACIL) which describes the physical facilities assigned to each course.
- o A Course data base (COURSE) which describes attributes and instructor assignments for each course.
- o An Instructor data base (INST) which contains personnel information and time utilization information for each instructor.

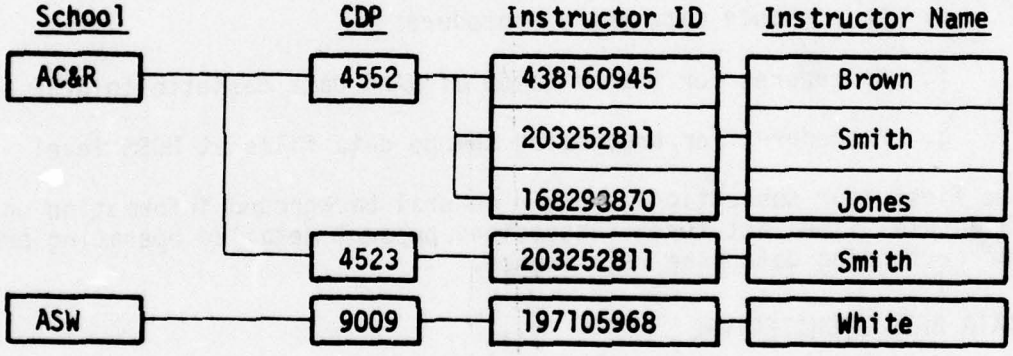
The three data bases support the DOTS TPF and SCRR models and provide data management/report generation capability to the User.

The data bases are implemented using RAMIS (Rapid Access Management Information System). RAMIS uses hierarchical (or "tree-structured") files. In addition to increasing storage and data maintenance efficiency, this structuring is conceptually most natural. Consider, for example, a very simple data base consisting of: (1) school code; (2) courses in each school (CDP numbers); and (3) instructors assigned to teach each course. A graphic comparison of tree-structured vs. unstructured design of this sample data base is shown below.

Unstructured (Sample Data Base)

School	CDP	Instructor ID	Instructor Name
AC&R	4552	438160945	Brown
AC&R	4552	203252811	Smith
AC&R	4552	168298870	Jones
AC&R	4523	203252811	Smith
ASW	9009	197105968	White

Tree-Structured (Sample Data Base)



The tree-structuring:

- Requires less storage. Higher level information need be stored only once rather than multiple times.
- Increases data management efficiency, since processing of redundant data is minimized or eliminated.

A "field" in a tree-structured file is a unit of information in the data base. In the sample data base. there are four fields: (1) school; (2) CDP; (3) instructor ID; and (4) instructor Name.

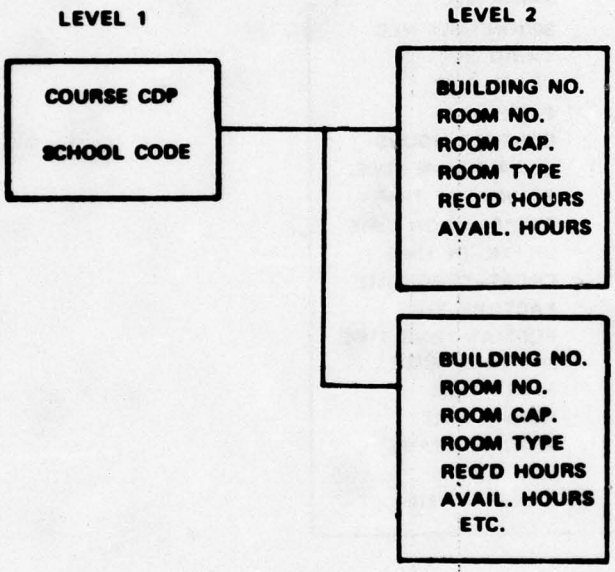
A "level" consists of all the fields in a single branch of the structure. In the sample data base, there are three levels: (1) school; (2) CDP; and (3) instructor ID and instructor name. The "highest" level is level 1; the "lowest" level (in the example above) is level 3.

A "segment" is a single occurrence of a group of data at a single level. For instance, in the file above the AC&R school has two subordinate CDP segments. For the same school, CDP segment 4552 controls three lower level segments; CDP 4523 controls a single level 3 segment.

One field within each segment is designated a "key field." The data content of this field serves to make the segment unique. In the sample data base above, school code would be key at the highest level. The level 2 key would be CDP number. For level 3, instructor ID would serve to distinguish each segment at that level.

STRUCTURES OF THE THREE DOTS DATA BASES

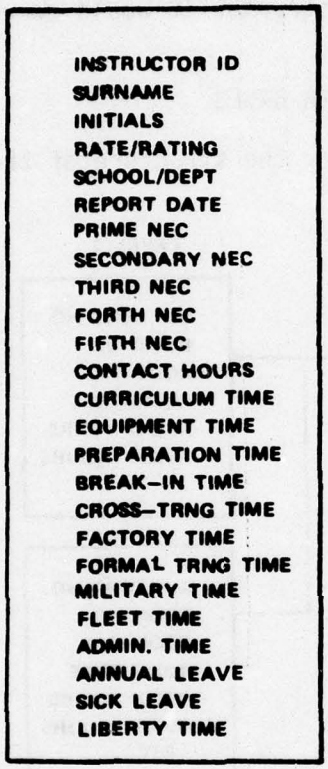
The following depicts pictorially the structure of the DOTS Facilities file (filename = FACIL).



DOTS FACILITIES FILE – HIERARCHIC STRUCTURE

Each level 1 segment may control several level 2 segments. The key field at the highest level (level 1) is course CDP number. The level 2 key is building number.

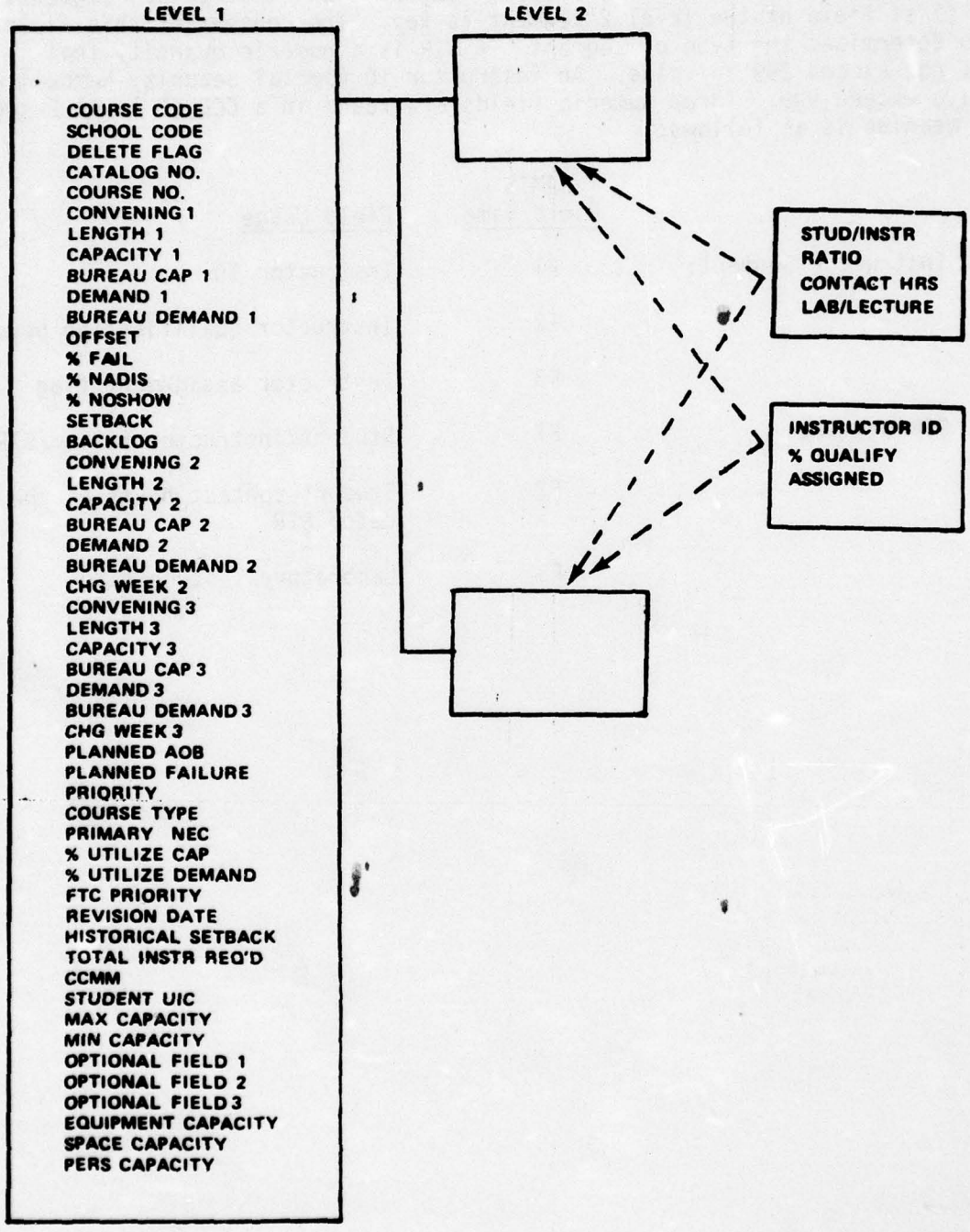
A pictorial representation of the DOTS Instructor file (filename = INST) follows. Note that this file essentially has no "hierarchic" structure since all of the data exists on a single level. The key field controlling the single high-level INST segment is instructor ID (Social Security Number).



DOTS INSTRUCTOR FILE – HIERARCHIC STRUCTURE

In the diagram following, the DOTS Course file (filename = COURSE) structure is shown. The high level key field is course CDP. The diagram indicates blank boxes for second level segments. This is intended to illustrate that the second level segment can be one of two distinct types. A level 2 segment contains three data fields. The use of these fields is shown in the two boxes to the far right.

DOTS COURSE FILE – HIERARCHIC STRUCTURE



In the discussions following, the level 2 segments of the COURSE file are called "student/instructor ratio"(SIR)" segments or "instructor" segments. The first field of the level 2 segment is key. The content of this field also determines the type of segment. A SIR is a numeric quantity that will not exceed 999 in value. An instructor ID (Social Security Number) will always exceed 999. Three numeric fields are found in a COURSE level 2 segment. The meaning is as follows:

	RAMIS Field Name	Field Usage
Instructor Segment:	F1	Instructor ID
	F2	Instructor qualification percentage
	F3	Instructor assignment flag
SIR Segment:	F1	Student/instructor ratio (SIR)
	F2	Student contact hours at the indicated SIR
	F3	Laboratory/lecture flag

DATA BASE DATA FIELD CONTENTS

As indicated in Figure I-1, an interactive program has been provided to enable the user to maintain the DOTS data base. All of the data items which are subject to periodic maintenance are listed in the following tables:

MAINTENANCE PROGRAM REFERENCE	RAMIS FIELD NAME	FORM	DEFINITION
BUILDING NUMBER	BLDG	AAAA	The name or number of the building in which a classroom or laboratory space is located.
ROOM NUMBER	RM	AAAA	The name or number of a classroom or laboratory space.
ROOM CAPACITY	RMCAP	NNN	The number of permanent seats, lab positions, trainer positions, etc., which can be utilized by students.
ROOM TYPE	RMPT	N	The type of training for which the space is utilized and is coded as follows: 1 - used for theory 2 - used for lab 3 - used for theory and lab 4 - designates a trainer
HOURS REQ'D PER CONV	REQHRS	NNNN.N	Required hours represent the number of hours the indicated space is required to convene one session of the referenced course.
TOT HR AVAIL/ YEAR	AVHRS	NNNNN	The number of hours, on an annual basis, the space is available for instructional purposes.

TABLE II-1

DOTS FACILITIES FILE DATA FIELDS

MAINTENANCE PROGRAM REFERENCE	RAMIS FIELD NAME	FORM	DEFINITION
INST ID(SSN)	ID	NNNNNNNNN	Social Security Number of Instructor.
LAST NAME	NME	A16	Instructor's last name.
FIRST, MIDDLE INITIALS	IN	AA	Instructor's initials.
RATE/RANK	RATE	AAAAA	Instructor's rate or rank; e.g., PN1 of LCDR (left justify).
REPORT DATE	REP	YYMM	The date the Instructor reported on board.
ROTATE DATE	ROT	YYMM	The Instructor's planned rotation date.
PRIMARY NEC	PNEC	NNNN	The instructors Primary NEC (if none, leave blank).
SECONDARY NEC	SNEC	NNNN	The Instructor's Secondary NEC (if none, leave blank).
ADDITIONAL NECS	NEC3 NEC4 NEC5	NNNN NNNN NNNN	Any additional NECS previously earned by the instructor. (If none, leave blank)
STUDENT CONTACT	CONT	NNNN	An estimate of the total number of hours per year expended on classroom contact with students.
CURRICULUM DEVELOPMENT	CDEV	NNNN	An estimate of the total number of hours per year expended on curriculum development.
CLASS PREP/ RELATED DUT	PREP	NNNN	An estimate of the total number of hours per year expended on class preparation and related duties.
INST TRNG (BREAK-IN)	BKIN	NNNN	An estimate of the total number of hours per year spent training to instruct a course, i.e., instructor break-in time.

TABLE II-2

DOTS INSTRUCTOR FILE DATA FIELDS

MAINTENANCE PROGRAM REFERENCE	RAMIS FIELD NAME	FORM	DEFINITION
EQUIP-TRNG/ MAINT	EQMT	NNNN	An estimate of the total number of hours per year expended on equipment maintenance or training to maintain equipment.
CROSS-TRNG (LOW STUD LOAD)	CTRN	NNNN	An estimate of the total number of hours per year expended on instructor cross-training due to low student load.
FORMAL INST SCH	ISCH	NNNN	An estimate of the total number of hours per year spent at the formal instructor training school.
FACTORY TRNG/ WKSHOPS	FTRN	NNNN	An estimate of the total number of hours per year expended on formal factory training or workshops.
SPEC FLEET ASSISTANCE	FLET	NNNN	An estimate of the total number of hours per year spent assisting the fleet with hardware or training problems not included in formal courses of instruction.
MIL DUT/PHYS TRNG/OTHER	MDUT	NNNN	An estimate of the total number of hours per year expended in the performance of military duties, physical training or related efforts.
SUPERVISION ADMIN	ADMN	NNNN	An estimate of the total number of hours per year expended on supervisory or administrative duties.
ANNUAL LEAVE	ANLV	NNNN	An estimate of the total number of hours per year spent on annual leave.
SICK LEAVE	SKLV	NNNN	An estimate of the total number of hours per year spent on sick leave.
SPECIAL LIBERTY	SLIB	NNNN	An estimate of the total number of hours per year spent on special liberty.

NOTE:

The total number of hours available per year is 2080. The above estimates of instructor activity should sum to 2080 hours.

TABLE II-2 (Cont'd)

MAINTENANCE PROGRAM REFERENCE	RAMIS FIELD NAME	FORM	DEFINITION
CDP NUMBER	CDP	AAAA	The CDP number is the unique 4 position alphanumeric course data processing code for each course taught in the Navy (NITRAS).
CIN NUMBER	CTLG	A-AAA-AAAA	The 10 position alphanumeric course Identification Number (CIN) used in the Catalog of Navy Training Courses (CANTRAC). This code is identical for the same course taught at different locations (NITRAS).
CRS SHORT TITLE	CRS	A16	The abbreviated descriptive title of the course (Course Short Title in NITRAS).
CONVEN	CONV1	NNN	The number of times the course is planned to convene during the current fiscal year. A judicious estimate of the number of classes to be convened should be used for courses which do not convene on scheduled dates (NITRAS).
LENGTH	LEN1	NNNN	The length in calendar days including weekend days, of the course for the current fiscal year (NITRAS).
CAP	CAP1	NNN	The normal class capacity is the maximum number of students that could be accommodated as a planned input each time a new class is convened for the course without detriment to the quality of instruction.
BCAP	BCAP1	NNN	The current number of seats in the course allocated to BUPERS or other agencies and not available for local quota control.
PLN INPUT	DMD1	NNNN	The current total anticipated annual demand or planned input for the course (NITRAS).
PLN B-INPUT	BDMD1	NNNN	The current annual demand or planned input for the course for students detailed by BUPERS.
CONVEN	CONV2	NNNN	Same as CONV1 but for FY + 1.

TABLE II-3

DOTS COURSE FILE DATA FIELDS

MAINTENANCE PROGRAM REFERENCE	RAMIS FIELD NAME	FORM	DEFINITION
LENGTH	LEN2	NNNN	Same as LEN1 but for FY + 1.
CAP	CAP2	NNN	Same as CAP1 but for FY + 1
BCAP	BCAP2	NNNN	Same as BCAP1 but for FY + 1.
PLAN INPUT	DMD2	NNNNN	Same as DMD1 but for FY + 1.
PLN B-INPUT	BDMD2	NNNNN	Same as BDMD1 but for FY + 1.
CONVEN	CONV3	NNN	Same as CONV1 but for FY + 2.
LENGTH	LEN3	NNNN	Same as LEN1 but for FY + 2.
CAP	CAP3	NNN	Same as CAP1 but for FY + 2.
BCAP	BCAP3	NNN	Same as BCAP1 but for FY + 2.
PLN INPUT	DMD3	NNNNN	Same as DMD1 but for FY + 2.
PLAN B-INPUT	BDMD3	NNNNN	Same as BDMD1 but for FY + 2.
FAILURE	PFAIL	NN.N%	The current historical (last 12 months) failure rate for the course.
NONAC DIS	NADIS	NN.N%	The historical (last 12 months) non-academic disenrollment rate for the course.
NO SHOW	NOSHOW	NN.N%	The historical (last 12 months) no-show rate experienced by the course. This rate is determined as the percentage of those scheduled for the course that did not attend or cancel.
SETBACK	SETB	NN.N%	The historical (last 12 months) set back rate for the course.
BACKLOG	BKLG	NNNN	The length in weeks that a student must wait for a local quota for a course.
PLANNED AOB	PAOB	NNNN	The planned average number of students on board. (Total student-days divided by 364)
PRIORITY CODE	PRTY	NN	The 2 digit Training Priority Designator code (NITRAS).

TABLE II-3 (Cont'd)

MAINTENANCE PROGRAM REFERENCE	RAMIS FIELD NAME	FORM	DEFINITION
COURSE TYPE	CTYPE	AA	The 2 position type code for the course indicating class of course/type training category (i.e., A1, C3, F2, etc.) (NITRAS).
NEC AWARDED	PNEC	NNNN	The primary NEC code for which the student is qualified by completion of this course. (NITRAS)
ACTIVITY/DEPT. CODE	SCH	NAAA	A code to identify the activity and the organization School/Department which conducts the course. The first position is coded as follows: 4 - NWTGP 5 - FITCPAC 6 - FLETRACEN 7 - FCDSTCP 8 - FASWTCP The remaining three positions can be any alphanumeric designation identifying the organizational unit which conducts the course.
STAFF UIC	SUIC	AAAA	The Staff Unit Identification Code (NITRAS)
COURSE UTILIZATION	UCAP	NNN%	The ratio of the total number of students enrolled in the course over a one year period to the total annual class capacity.
DATE LAST COURSE REVIEW	RDTE	YYMMDD	The date on which the course curriculum was last reviewed.
CCMM	CCMM	AAAAAAAAAA	The activity assigned course curricula model manager responsibility.
CLASS CAPACITY MAX	MAXCAP	NNN	The maximum number of quotas which can be grated for any one class. This figure is used for quota control purposes and may be equal to or greater than the normal class capacity.
CLASS CAPACITY MIN	MINCAP	NNN	The minimum number of quotas which will be granted for any one class. The class will not be convened if the number of quotas requested is less than the number.

TABLE II-3 (Cont'd)

MAINTENANCE PROGRAM REFERENCE	RAMIS FIELD NAME	FORM	DEFINITION
CLASS CAPACITY (PERS)	PCAP	NNN	The maximum number of students that may be trained per class based on the number of authorized billets. The number entered assumes that unlimited space and equipment are available.
CLASS CAPACITY (EQUIP)	ECAP	NNN	The maximum number of students that may be trained per class based on available equipment capability only. The number entered assumes that unlimited personnel and space are available.
CLASS CAPACITY (SPACE)	SCAP	NNN	The maximum number of students that may be trained per class based on the availability of assigned training space. The number entered assumes that unlimited personnel and equipment are available.
OPTIONAL 1	OPT1	AAAAAA	Any six character alphanumeric data may be stored in this field at the discretion of the individual activity.
OPTIONAL 2	OPT2	AAAAAA	Any six character alphanumeric data may be stored in this field at the discretion of the individual activity.
OPTIONAL 3	OPT3	AAAAAA	Any six character alphanumeric data may be stored in this field at the discretion of the individual activity.
INST ID(SSN)	F1	NNNNNNNNN	Social Security number of instructor who is qualified to teach this course (CDP).
% QUALIFIED	F2	NNN	The portion of the total curriculum that the instructor is qualified to teach.
ASSIGN FLAG	F3	N	A 1 in this field indicates that the instructor is currently assigned to teach this course. A Ø indicates that the instructor is qualified but is not currently assigned to teach the course.
STUDENT/ INSTRUCTOR RATIO	F1	NNN	The ratio of students to each instructor. A student/instructor ratio of 25 to 1 indicates a ratio of 25 students for one instructor. Since each facet of training associated with the course may require a different ratio of students to each instructor, several ratios may be entered for each course.

TABLE II-3 (Cont'd)

MAINTENANCE PROGRAM REFERENCE	RAMIS FIELD NAME	FORM	DEFINITION
CONTACT HOURS AT THIS RATIO	F2	NNNN	The number of instructional contact hours taught at a given ratio of trainees per instructor. A contact hour represents sixty minutes of instruction. This refers to clock hours of curriculum time devoted to actual instruction, exclusive of breaks, administrative time, lunch, medical, dental, etc. The total instructional contact hours taught at the various ratios must equal the total contact hours on the Master Schedule.
RATIO TYPE	F3	N	Differentiates between theory and laboratory instruction a 1 indicates lecture/ theory instruction; 2 indicates laboratory instruction; 3 indicates a combination of theory and lab instruction at the same student/instructor ratio.

TABLE II-3 (Cont'd)

GENERAL MAINTENANCE PROCEDURES

The DOTS Data Base presently structured under RAMIS requires periodic maintenance as a result of changing data on courses, personnel, and facility resources. These procedures assume that maintenance will be accomplished by some centralized staff function at each activity. Sufficient front-end editing is performed to ensure data integrity and an audit trail is maintained to permit rapid identification of erroneous inputs. Changes are made interactively with sufficient prompting to reduce reference by the user to written maintenance procedures.

The following general approach will be used for adding, deleting, and changing data items within RAMIS.

A. Activity Level

Data base listings will be red-lined to denote changes by the appropriate division department or staff office at the activity level. A trained individual, with the aid of a local computer-driven prompting display, will input the change data. Local editing and correcting will be performed and a change tape file will be created. The change tape file will be periodically transmitted to the functional level computer.

B. Functional Level

Change requests transmitted to the functional level will be subjected to another level (manual) of editing and correcting. The Data Base Manager (DBM) will monitor change activity by reviewing change tape listings and will intervene when activities have submitted changes which are questionable or erroneous. The DBM will have the capability of editing the activity change tapes. A listing of the edited version of the activity change tape will be returned to each activity. The DBM will be conversant with RAMIS procedures and capable of correcting proposed data base changes which are rejected by the RAMIS editing procedures.

C. NCSS Level

Any additional editing not possible at the activity or functional level will be accomplished at the NCSS level. A RAMIS error report will be generated subsequent to processing the proposed changes through TAMIS. The identified errors must be examined and corrected at either the functional or the activity level. The NCSS error file will then be edited by the DBM and be resubmitted to RAMIS.

Change Data Processing: The specific data processing sequence for each of the steps in the change (maintenance) procedure (see Figure II-1) are defined below.

1. Division/department levels will note changes as they occur by red-lining copies of the data base listing.

2. An interactive prompting program will assist a trained user to input changes into the activity's computer. (See operating procedures under the next subheading.)

3. The activity level data entry program will check inputs for proper type; e.g., numeric, and length of data elements.

4. Changes successfully passing the editing process will be used to create a change file stored on a WANG tape cassette.

5. The change tape file will be transmitted (or mailed) to the functional level on a periodic basis. Information to identify school, department, date, etc., should be written on the tape cassette label.

6. Upon receipt of the activity change tape, the DBM will obtain a hard-copy listing of the tape and will manually verify the submitted changes.

7. The DBM will review all data base change exceptions and take appropriate action, which might include:

   Overriding an activity's input
   Manual verification of exception with activity
   Return of input to activity without processing
   Removal of input from change file and manual input of change
   Acceptance of change for inclusion in the change file

8. All changes successfully passing the edit step and the action by the DBM will be incorporated into a change file for subsequent processing into RAMIS. The change file will be transmitted to the NCSS environment using the procedures outlined in this section. A listing of the revised change tape will be printed prior to the RAMIS processing step. This listing should be returned to the activities upon successful completion of the RAMIS change processing. Receipt of the listing will indicate that the changes listed have been incorporated into the data base.

9. The DBM will initiate the RAMIS change file processing in the NCSS environment. The DBM will be responsible for correcting any additional errors identified by the RAMIS data base edit procedures. These corrections should be noted on the revised change file listing, and then be reentered into RAMIS. After all changes have been incorporated into the RAMIS files, new data base listings will be printed and forwarded to each of the activities.

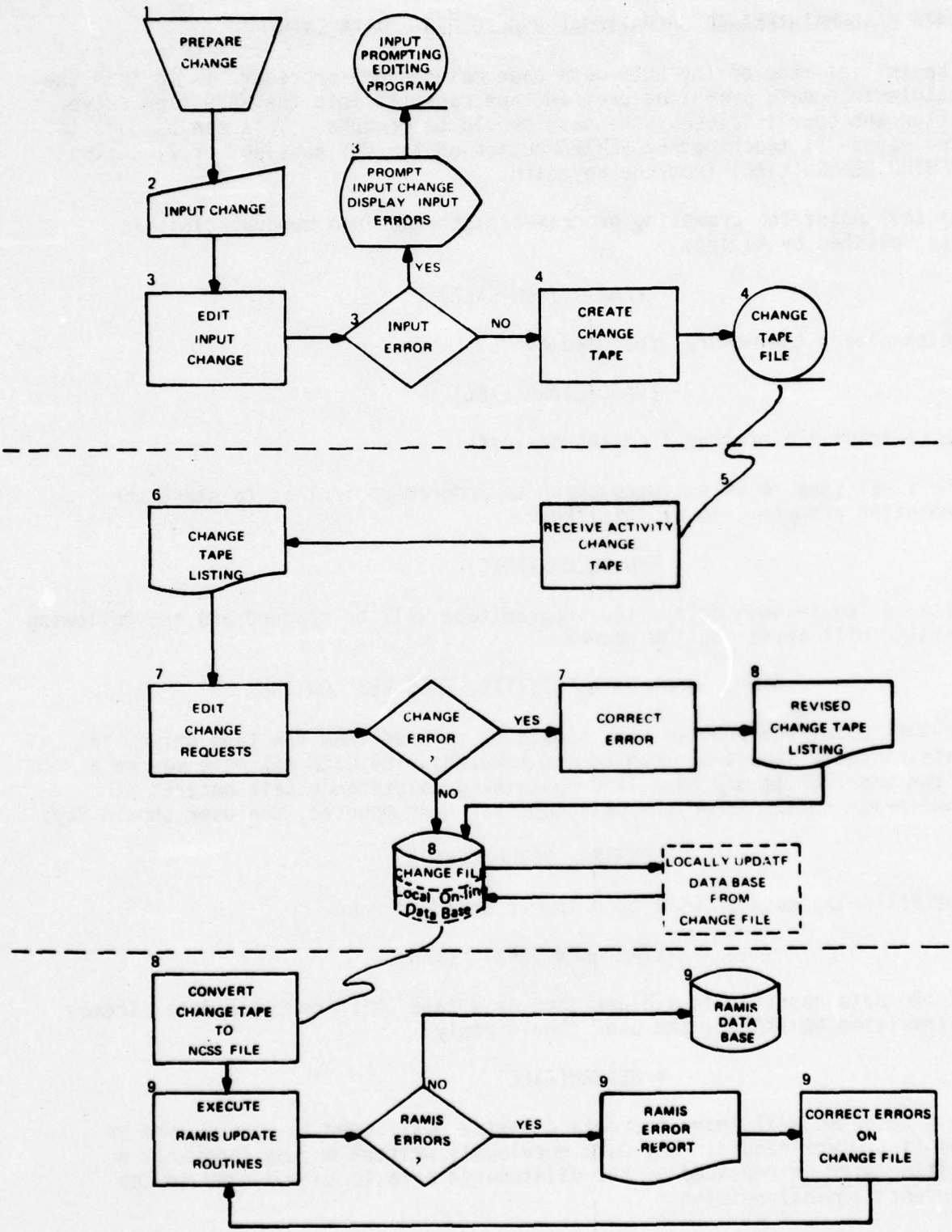


FIGURE II-1

DATA BASE MAINTENANCE PROCEDURE

DATA BASE MAINTENANCE OPERATIONAL PROCEDURES - DATA ENTRY

The initial step of the DOTS data base maintenance procedure is to load the maintenance data prompting program tape cassette into the WANG tape drive. After the door is closed, the tape should be rewound. This can be done in two ways: 1) touching the REWIND button on the CRT housing, or 2) keying REWIND RETURN(EXEC) from the keyboard.

At this point the prompting program can be read into memory. This is accomplished by keying:

CLEAR RETURN(EXEC)

which clears the memory, followed by

LOAD RETURN(EXEC)

which loads the program from the cassette.

The final step to be executed prior to program control is to start the prompting program. To do this, key:

RUN RETURN(EXEC)

After a few seconds delay, the program tape will be rewound and the following message will appear on the screen:

STOP; MOUNT DATA CASSETTE; THEN KEY CONTINUE

At this point the program tape should be removed from the tape drive. A data cassette should be mounted and rewound. The data cassette may be a blank tape, or it may be a tape containing maintenance data entered at some prior time. After the data tape has been mounted, the user should key:

CONTINUE RETURN(EXEC)

The following message will then appear on the screen:

START NEW TAPE? (Y/N)

If the data cassette is a blank tape or a tape which contains data already transmitted to TRAPAC, the user should reply:

Y RETURN(EXEC)

This response will leave the data cassette positioned to record data on the first tape record. Any data previously written on the tape will be written over or replaced by the maintenance data to be recorded in the current terminal session.

If the data cassette contains maintenance data which has not been transmitted to TRAPAC, and therefore should not be erased, the user should reply:

N RETURN(EXEC)

This response will position the data cassette to record immediately following the last maintenance data recorded. The user will then be able to append current data to data already recorded on the tape.

After the data cassette has been properly positioned, the DOTS Data Base Maintenance Menu will appear on the screen:

DOTS DATA BASE MAINTENANCE MENU

	ADD	CHANGE	DELETE
CDP NUMBER	21	16	11
ACTIVITY/DEPT NO.	*	20	*
COURSE DATA	*	6	*
STUD/INST RATIO	23	17	12
CONTACT HOURS	*	7	*
INSTRUCTOR ID	24	18	13
INSTRUCTOR DATA	*	3	*
INSTRUCTOR ASSGNMT	26	8	27
BLDG/ROOM NUMBER	25	19	14
BLDG/ROOM DATA	*	2	*

KEY NUMBER INDICATING ACTION TO BE TAKEN.
(IF NO ADDITIONAL MAINTENANCE REQUIRED, KEY RETURN)

After all maintenance transactions have been entered, the user should key RETURN(EXEC) in response to the instruction at the bottom of the maintenance MENU. The following message will appear on the screen:

DATA TAPE COMPLETE? (Y/N)

If the data cassette is only partially full and the user wishes to add more maintenance transactions to the same cassette at some time in the fugure, key

N

If no more maintenance transactions are to be added to the tape and the tape is to be sent to COMTRAPAC for data base processing, key:

Y

The following message is then displayed:

END PROGRAM
FREE SPACE XXXXX

and program execution terminates. XXXXX is the approximate amount of memory (in bytes) not used by the program.

The final step is to remove the data cassette from the WANG tape drive unit and label the cassette with the activity name and date. The data cassette will be rewound automatically prior to program execution termination.

DOTS DATA BASE COURSE FILE MAINTENANCE

The Course file contains course data (see Course file listing, Section A) such as number of convenings, capacity, length, input, SUIC, CCMM; student/instructor ratio data (see Course file listing, Section B); and instructor assignment data (see Course file listing, Section C).

Data base changes will be recorded as they occur by red-lining current listings of the data base contents. The red-lined listings will be the source document for recording changes on the data cassette. A sample DOTS Course file listing is shown below:

DOTS COURSE FILE LISTING (FILENAME = COURSEF) PAGE : 16
ON 08/09/78 AT 1437

CUR	SCH	CTLG	CNS	CTYPE	PNEC	SUIC	NOTE	CCMM	PRTY	FPRTY				SIR/ ID	CHRS/ PQUAL	TY/ ASN
														504667850	100	1
9362	SINT	J= 3A-0955	BASIC INTEL MES	F2		0388A	770106	FITCLANT	7					0	0	0
														5	5	0
	CONV1 :	4	CONV2 : 4	CONV3 :	4	MAXCAP :	24	PFAIL :	5.000	MSETR :	.000			10	5	0
	LEN1 :	12	LEN2 : 12	LEN3 :	12	MINCAP :	5	NADIS :	.020	TINSTR :	.5			25	50	0
	CAP1 :	20	CAP2 : 20	CAP3 :	20	PCAP :	20	NOSHOW :	.050	UDFM :	0 0			138365210	100	1
	MCAP1 :	0	MCAP2 : 0	MCAP3 :	0	FCAP :	24	SETR :	.000	OPT1 :				277628748	100	1
	DMD1 :	74	DMD2 : 74	DMD3 :	74	SCAP :	24	BKLG :	0	OPT2 :				323647393	100	1
	BDMD1 :	0	BDMD2 : 0	BDMD3 :	0	UCAP :	84	PAOR :	0	OPT3 :				335404714	100	1
														441428831	100	1
														457923645	100	1
														539463563	100	1
														541500004	100	1
														504667850	100	1
9363	SINT	J= 3A-0955	AIM INTEL MES	F2		0388A	770106	FITCLANT	7					0	0	0
														10	43	0
	CONV1 :	4	CONV2 : 4	CONV3 :	4	MAXCAP :	24	PFAIL :	4.000	MSETR :	.000			25	25	0
	LEN1 :	12	LEN2 : 12	LEN3 :	12	MINCAP :	5	NADIS :	.000	TINSTR :	.5			152308689	100	1
	CAP1 :	20	CAP2 : 20	CAP3 :	20	PCAP :	20	NOSHOW :	.050	UDFM :	0 0			277628748	100	1
	MCAP1 :	0	MCAP2 : 0	MCAP3 :	0	FCAP :	20	SETR :	.000	OPT1 :				335404714	100	1
	DMD1 :	56	DMD2 : 56	DMD3 :	56	SCAP :	5	BKLG :	0	OPT2 :				441428831	100	1
	BDMD1 :	0	BDMD2 : 0	BDMD3 :	0	UCAP :	64	PAOR :	0	OPT3 :				539463563	100	1
9364	SINT	J= 3A-0956	IMAGE INTERP MES	F2		0388A	770106	FITCLANT	7					0	0	0
														6	28	0
	CONV1 :	5	CONV2 : 5	CONV3 :	5	MAXCAP :	24	PFAIL :	3.000	MSETR :	.000			25	40	0
	LEN1 :	12	LEN2 : 12	LEN3 :	12	MINCAP :	5	NADIS :	.000	TINSTR :	2.9			306666540	100	1
	CAP1 :	20	CAP2 : 20	CAP3 :	20	PCAP :	0	NOSHOW :	.050	UDFM :	0 0			323647393	100	1
	MCAP1 :	0	MCAP2 : 0	MCAP3 :	0	FCAP :	0	SETR :	.000	OPT1 :				441428831	100	1
	DMD1 :	85	DMD2 : 85	DMD3 :	85	SCAP :	0	BKLG :	0	OPT2 :				539463563	100	1
	BDMD1 :	0	BDMD2 : 0	BDMD3 :	0	UCAP :	64	PAOR :	0	OPT3 :						
9365	SINT	J= 3A-5014	NAV INT FUNC OFF	C2		0388A	770331	FITCPAC	0					0	0	0
														3	7	0
	CONV1 :	1	CONV2 : 1	CONV3 :	1	MAXCAP :	16	PFAIL :	.000	MSETR :	.000			4	12	0
	LEN1 :	56	LEN2 : 56	LEN3 :	56	MINCAP :	5	NADIS :	.000	TINSTR :	.4			6	104	0
	CAP1 :	16	CAP2 : 16	CAP3 :	16	PCAP :	16	NOSHOW :	.100	UDFM :	0 0			25	104	0
	MCAP1 :	0	MCAP2 : 0	MCAP3 :	0	FCAP :	16	SETR :	.000	OPT1 :				138365210	100	1
	DMD1 :	8	DMD2 : 8	DMD3 :	8	SCAP :	16	BKLG :	0	OPT2 :				152308689	100	1
	BDMD1 :	0	BDMD2 : 0	BDMD3 :	0	UCAP :	24	PAOR :	0	OPT3 :				277628748	100	1
														306666540	100	1
														323647393	100	1

A B C

MAINTENANCE TRANSACTION NUMBER: 6

TRANSACTION NAME: Change Course Data

DATA SOURCE: Course File Listing - Section A

FUNCTION: Change the data contained in any data field listed in Section A of the Course file listing (except the course CDP number).

KEYIN REQUIRED: 6 RETURN(EXEC) (From MENU)

RESPONSE:

*CHANGE COURSE DATA FOR THE FOLLOWING COURSE

CDP NUMBER ----

ACTIVITY/DEPT CODE ----
(IF SAME AS PREVIOUS TRANSACTION (NNNN),KEY RETURN.)

NOTES: 1) Typing errors may be corrected by backspacing over error and retyping.

2) Data is not stored until RETURN(EXEC) is keyed.

3) Dashes indicate the maximum number of characters which can be entered. It is not necessary to enter a character in each space.

4) If the activity/dept code to be entered is the same as the previous entry, it can be entered automatically by keying RETURN(EXEC).

TO TERMINATE PAGE: Key RETURN(EXEC) after entering Activity/Dept code.

RESPONSE: TRANS 6 page 2 (see next page).

PRECEDING PAGE BLANK-NOT FILMED

MAINTENANCE TRANSACTION NUMBER: 6 (Cont'd)

TRANSACTION NAME: Change Course Data

DATA SOURCE: Course File Listing - Section A

FUNCTION: Change the data contained in any data field listed in Section A of the Course file listing (except the course CDP number).

KEYIN REQUIRED: RETURN(EXEC) (From TRANS 6 page 1)

RESPONSE:

```
          ********** COURSE CDP # NNNN **********
01 CURRENT YEAR: CONVEN/LENGTH/CAP/BCAP/PLN INPUT/PLN B-INPUT
                 ___   /_____  /____/ ___/_____   /_____
02 CUR YEAR  +1: CONVEN/LENGTH/CAP/BCAP/PLN INPUT/PLN B-INPUT
                 ___   /_____  /___/___/_____   /_____
03 CUR YEAR  +2: CONVEN/LENGTH/CAP/BCAP/PLN INPUT/PLN B-INPUT
                 ___   /_____  /___/___/_____   /_____
04 STUDENT CLASS CAPACITIES:
       MAX=___  MIN=____ PERS=___  EQUIP=___  SPACE=___
05 HISTORICAL CRS RATES:
   FAILURE=__.-%  NONAC DIS=__.-%  NOSHOW=__.-%  SETBACK=__.-%

KEY LINE NO CONTAINING DATA TO BE ENTERED OR CHANGED.
TO SKIP A DATA ITEM ON A LINE KEY RETURN; CURSOR WILL ADVANCE TO
THE NEXT DATA ITEM. TO EDIT REMAINING COURSE DATA, RETURN -
```

NOTES: 1) Typing errors may be corrected by backspacing over error and retyping.

2) Data is not stored until RETURN(EXEC) is keyed.

3) Dashes indicate the maximum number of characters which can be entered. It is not necessary to enter a character in each space.

4) Only those data items for which the data has changed need be entered. To change a single data item in a line containing several items, key RETURN(EXEC) for those items which have not changed.

5) A line number may be entered more than one time to change a data item that may have been overlooked. Key the line number in response to the instruction at the bottom of the page.

6) Historical course rates are shown on the Course file listing as decimal numbers, i.e., .030, which is 3%. To enter a 3% failure rate in line 5, keyin 0 3 . 0%.

TO TERMINATE PAGE: Key RETURN(EXEC).

RESPONSE: TRANS 6 page 3 (see next page).

MAINTENANCE TRANSACTION NUMBER: 6 (Cont'd)

TRANSACTION NAME: Change Course Data

DATA SOURCE: Course File Listing - Section A

FUNCTION: Change the data contained in any data field listed in Section A of the Course file listing (except the course CDP number).

KEYIN REQUIRED: RETURN(EXEC) (From TRANS 6 page 2)

RESPONSE:

```
********** COURSE CDP # NNNN **********
06 ACT/DEPT=----  CIN=---------- CRS TITLE=------------------

07 COURSE UTILIZATION=---%  PLANNED AOB=----  BACKLOG=----

08 DATE NEXT COURSE REVIEW=YY/MM/DD     STAFF UIC=----

09 CCMM=----------  PRIORITY CODE =--

10 COURSE TYPE=--  NEC AWARDED=----

11 OPTIONAL1=------  OPTIONAL2=------  OPTIONAL3=------
KEY LINE NO CONTAINING DATA TO BE ENTERED OR CHANGED.
TO SKIP A DATA ITEM ON A LINE KEY RETURN; CURSOR WILL ADVANCE TO
THE NEXT DATA ITEM. IF NO MORE EDITING REQ'D,KEY RETURN -
```

NOTES: 1) Typing errors may be corrected by backspacing over error and retyping.

2) Data is not stored until RETURN(EXEC) is keyed.

3) Dashes indicate the maximum number of characters which can be entered. It is not necessary to enter a character in each space.

4) Data for data items on this page may be entered or skipped as required.

TO TERMINATE PAGE: Key RETURN(EXEC).

RESPONSE: MENU

MAINTENANCE TRANSACTION NUMBER: 7

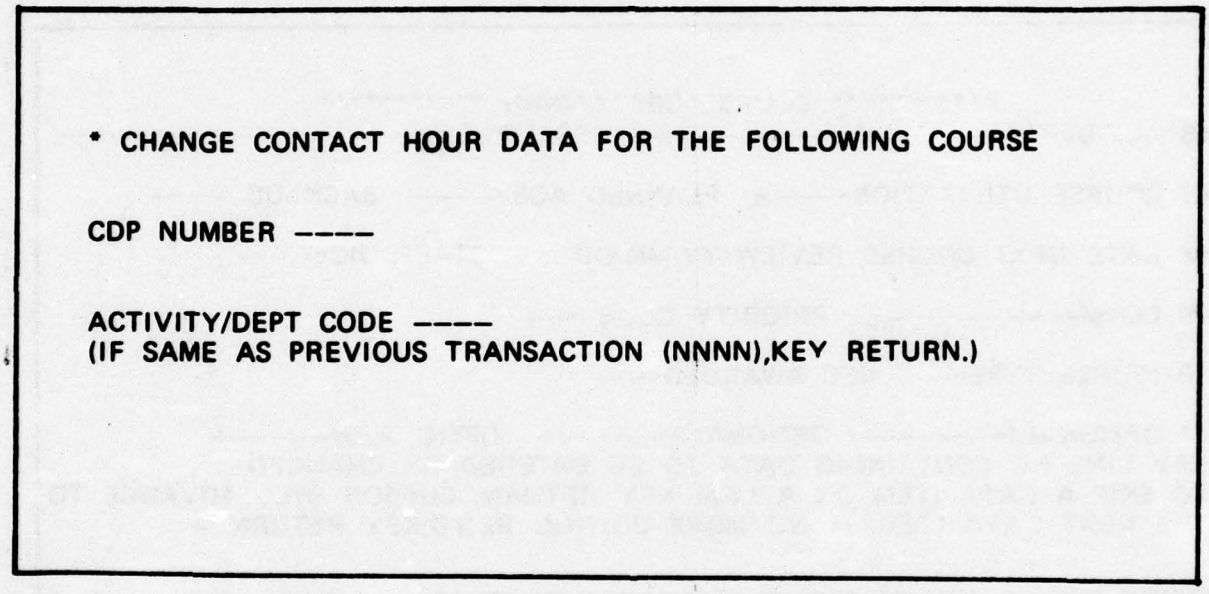
TRANSACTION NAME: Change Contact Hour Data

DATA SOURCE: Course File Listing - Section B

FUNCTION: Change either the contact hour or the course type data for a specified student/instructor ratio

KEYIN REQUIRED: 7 RETURN(EXEC) (From MENU)

RESPONSE:



NOTES: 1) Typing errors may be corrected by backspacing over error and retyping.

2) Data is not stored until RETURN(EXEC) is keyed.

3) Dashes indicate the maximum number of characters which can be entered. It is not necessary to enter a character in each space.

4) If the activity/dept code to be entered is the same as the previous entry, it can be entered automatically by keying RETURN(EXEC).

TO TERMINATE PAGE: Key RETURN(EXEC) after entering activity/dept code.

RESPONSE: TRANS 7 page 2 (see next page).

MAINTENANCE TRANSACTION NUMBER: 7 (Cont'd)

TRANSACTION NAME: Change Contact Hour Data

DATA SOURCE: Course File Listing - Section B

FUNCTION: Change either the contact hour or the course type data for a specified student/instructor ratio.

KEYIN REQUIRED: RETURN(EXEC) (From TRANS 7 page 1)

RESPONSE:

```
CDP#NNNN

   STUDENT/INSTRUCTOR RATIO= ---/1

   CONTACT HOURS AT THIS RATIO = ----

   RATIO TYPE = -
    1=LECTURE
    2=LAB
    3=COMB LEC/LAB

TO ENTER ADDITIONAL CONTACT HOUR CHANGES FOR THE ABOVE
COURSE TYPE 1, ELSE KEY RETURN-
```

NOTES: 1) Typing errors may be corrected by backspacing over error and retyping.

2) Data is not stored until RETURN(EXEC) is keyed.

3) Dashes indicate the maximum number of characters which can be entered. It is not necessary to enter a character in each space.

4) The student/instructor ratio must be entered. The contact hour and ratio type data are optional. To skip a data item key RETURN(EXEC); the cursor will be advanced to the next data item.

5) To change data associated with other student/instructor ratios for the same course, key 1 RETURN(EXEC) in response to the instruction at the bottom of the page.

TO TERMINATE PAGE: Key RETURN(EXEC)

RESPONSE: MENU

MAINTENANCE TRANSACTION NUMBER: 8

TRANSACTION NAME: Change Instructor Assignment Data

DATA SOURCE: Course File Listing - Section C

FUNCTION: Change the data in either the percent qualified field or the assign flag field

KEYIN REQUIRED: 8 RETURN(EXEC) (From MENU)

RESPONSE: -

```
CHANGE ASSIGNMENT DATA FOR THE FOLLOWING INSTRUCTOR:

INSTRUCTOR ID(SSN) =___-__-____
CDP NUMBER ____
ACTIVITY/DEPT CODE ____
(IF SAME AS PREVIOUS TRANSACTION (NNNN), KEY RETURN.)

NEW ASSIGNMENT DATA:
% QUALIFIED = ___%

ASSIGN FLAG = _
(ENTER 0 IF NOT TEACHING ABOVE COURSE; ELSE ENTER 1)

TO ENTER ADDITIONAL CHANGES FOR THE ABOVE
INSTRUCTOR TYPE 1; ELSE KEY RETURN -
```

NOTES: 1) Typing errors may be corrected by backspacing over error and retyping.

2) Data is not stored until RETURN(EXEC) is keyed.

3) Dashes indicate the maximum number of characters which can be entered. It is not necessary to enter a character in each space.

4) Data must be entered for instructor ID, CDP number, and activity/dept code. Data for percent qualified and assign flag are optional. Data for these fields may be skipped by keying RETURN(EXEC).

5) Assignment data changes for the same instructor but for additional courses can be made by keying 1 RETURN(EXEC) in response to the instruction at the bottom of the page.

TO TERMINATE PAGE: Key RETURN(EXEC)

RESPONSE: MENU

MAINTENANCE TRANSACTION NUMBER: 11

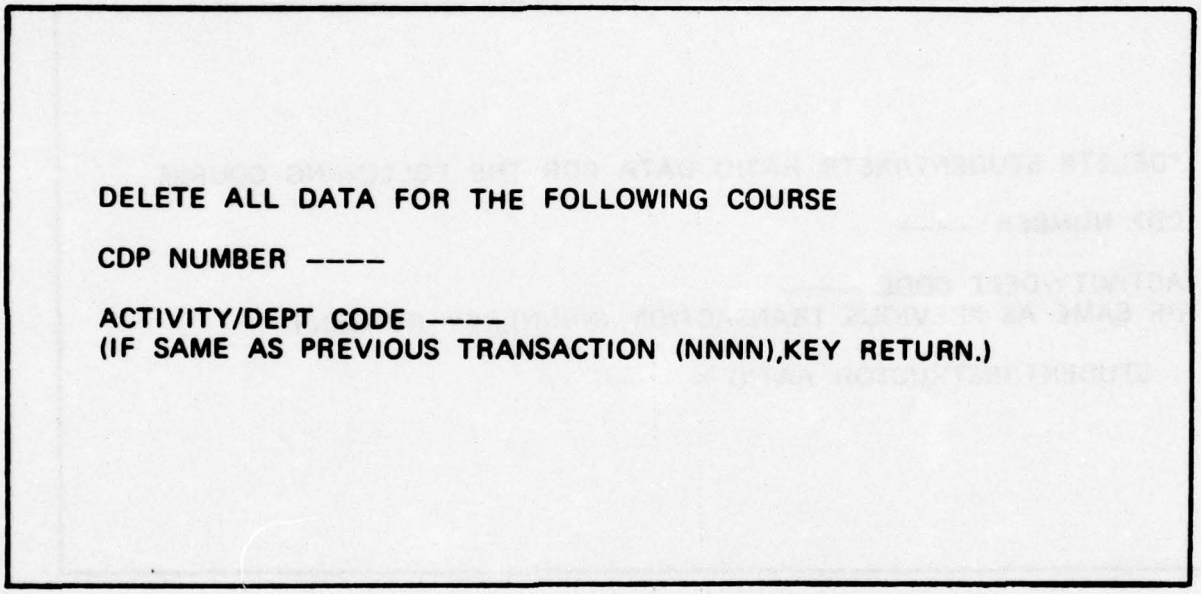
TRANSACTION NAME: Delete a CDP Number

DATA SOURCE: Course File Listing; Facility File Listing

FUNCTION: Delete all data associated with the specified CDP number in both the Course file and the Facility file

KEYIN REQUIRED: 11 RETURN(EXEC) (From MENU)

RESPONSE:



NOTES: 1) Typing errors may be corrected by backspacing over error and retyping.

2) Data is not stored until RETURN(EXEC) is keyed.

3) Dashes indicate the maximum number of characters which can be entered. It is not necessary to enter a character in each space.

TO TERMINATE PAGE: Key RETURN(EXEC) after entering activity/dept code.

RESPONSE: MENU

TAEG REPORT NO. 36

MAINTENANCE TRANSACTION NUMBER: 12

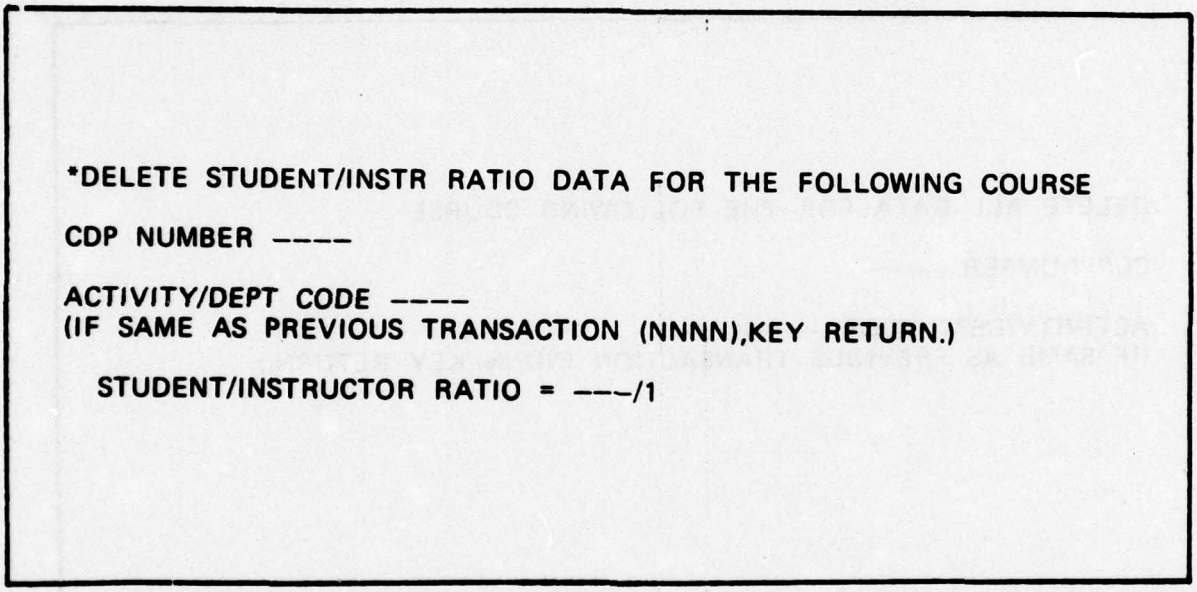
TRANSACTION NAME: Delete Student/Instructor Ratio

DATA SOURCE: Course File Listing - Section B

FUNCTION: Delete a student/instructor ratio and all associated data from the specified course

KEYIN REQUIRED: 12 RETURN(EXEC) (From MENU)

RESPONSE:



NOTES: 1) Typing errors may be corrected by backspacing over error and retyping.

2) Data is not stored until RETURN(EXEC) is keyed.

3) Dashes indicate the maximum number of characters which can be entered. It is not necessary to enter a character in each space.

4) If the activity/dept code to be entered is the same as the previous entry, it can be entered automatically by keying RETURN(EXEC).

TO TERMINATE PAGE: Key RETURN(EXEC) after entering student/instructor ratio.

RESPONSE: MENU

MAINTENANCE TRANSACTION NUMBER: 16

TRANSACTION NAME:	Change a CDP Number
DATA SOURCE:	Course File Listing - Section A; Facility File Listing
FUNCTION:	Change the course CDP number from the current value to a new value
KEYIN REQUIRED:	16 RETURN(EXEC) (From MENU)
RESPONSE:	

```
    * CHANGE THE FOLLOWING CDP NUMBER

OLD CDP NUMBER ----

NEW CDP NUMBER ----

     ACTIVITY/DEPT CODE ----
     (IF SAME AS PREVIOUS TRANSACTION (NNNN),KEY RETURN.)
```

NOTES: 1) Typing errors may be corrected by backspacing over error and retyping.

2) Data is not stored until RETURN(EXEC) is keyed.

3) Dashes indicate the maximum number of characters which can be entered. It is not necessary to enter a character in each space.

4) This transaction will change the CDP number in both the Course file and the Facility file.

5) All data changes in both the Course file and the Facilities file associated with a course for which the CDP number has been changed should refer to the new CDP number.

TO TERMINATE PAGE: Key RETURN(EXEC) after entering Activity/Dept code.

RESPONSE: MENU

TAEG REPORT NO. 36

MAINTENANCE TRANSACTION NUMBER: 17

TRANSACTION NAME: Change the Student/Instructor Ratio

DATA SOURCE: Course File Listing - Section B

FUNCTION: Change the student/instructor ratio from the current value to a new value

KEYIN REQUIRED: 17 RETURN(EXEC) (From MENU)

RESPONSE:

*CHANGE STUDENT/INSTR RATIO DATA FOR THE FOLLOWING COURSE

CDP NUMBER ----

ACTIVITY/DEPT CODE ----
(IF SAME AS PREVIOUS TRANSACTION (NNNN),KEY RETURN.)

OLD STUDENT/INSTRUCTOR RATIO = ----/1

NEW STUDENT/INSTRUCTOR RATIO = ----/1

NOTES: 1) Typing errors may be corrected by backspacing over error and retyping.

2) Data is not stored until RETURN(EXEC) is keyed.

3) Dashes indicate the maximum number of characters which can be entered. It is not necessary to enter a character in each space.

4) This transaction is used to change the value of a student/instructor ratio. To change contact hour or course type data use transaction 7. If both the ratio and either contact hour or course type data are changed using transactions 17 and 7, the new student/instructor ratio should be used when entering the data for transaction 7.

5) An alternate approach to using both transactions 17 and 7 is to delete the current ratio (transaction 12) and then add a new ratio (transaction 23) which will include both contact hour and course type.

TO TERMINATE PAGE: Key RETURN(EXEC) after entering new student/instructor ratio.

RESPONSE: MENU

MAINTENANCE TRANSACTION NUMBER: 20

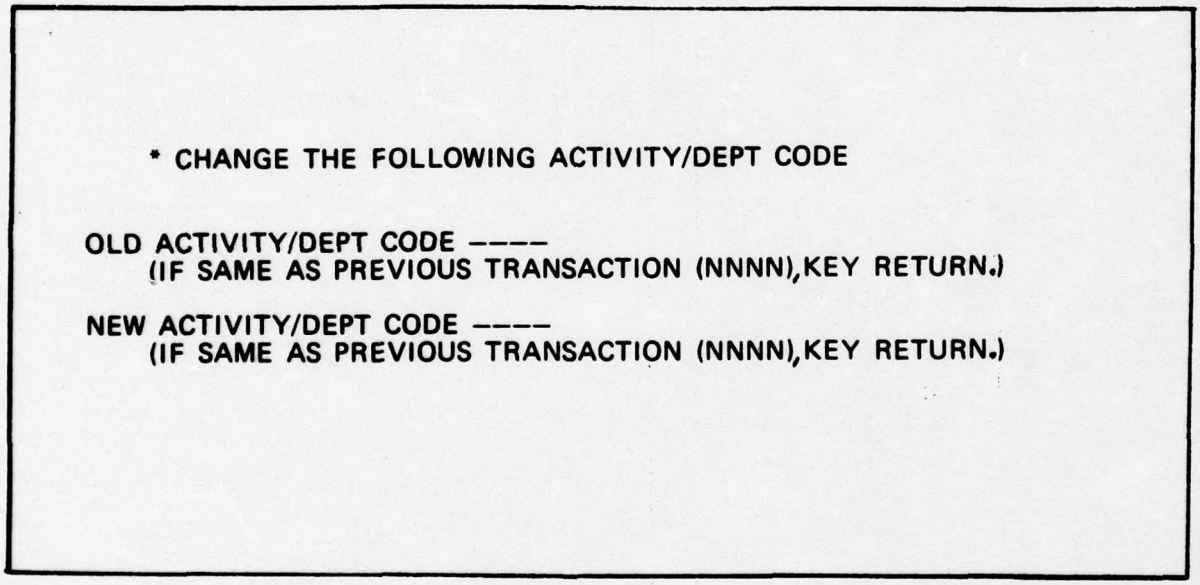
TRANSACTION NAME: Change Activity/Dept Number

DATA SOURCE: Course File Listing; Instructor File Listing; Facility File Listing

FUNCTION: Change the activity/dept code from the current value to a new value for all occurrences in the Course, Instructor, and Facility files

KEYIN REQUIRED: 20 RETURN(EXEC) (From MENU)

RESPONSE:



NOTES: 1) Typing errors may be corrected by backspacing over error and retyping.

2) Data is not stored until RETURN(EXEC) is keyed.

3) Dashes indicate the maximum number of characters which can be entered. It is not necessary to enter a character in each space.

4) This transaction will change the activity/dept code on each course record in the Course file, each instructor record in the Instructor file, and each course record in the Facility file.

TO TERMINATE PAGE: Key RETURN(EXEC) after entering new activity/dept code.

RESPONSE: MENU

MAINTENANCE TRANSACTION NUMBER: 21

TRANSACTION NAME: Add a CDP Number

DATA SOURCE: Course File Listing - Section A

FUNCTION: Add a new course and associated data to the course file

KEYIN REQUIRED: 21 RETURN(EXEC) (From Menu)

RESPONSE:

```
* ADD THE FOLLOWING COURSE

CDP NUMBER.....----

CIN NUMBER.....----------

CRS SHORT TITLE.....----------------

ACTIVITY/DEPT CODE.----
```

NOTES: 1) Typing errors may be corrected by backspacing over error and retyping.

2) Data is not stored until RETURN(EXEC) is keyed.

3) Dashes indicate the maximum number of characters which can be entered. It is not necessary to enter a character in each space.

4) Acceptable CIN numbers must begin with one of the following characters: A,J,K,S,X.

TO TERMINATE PAGE: Key RETURN(EXEC) after entering Activity/Dept code.

RESPONSE: TRANS 21 page 2 (see next page).

MAINTENANCE TRANSACTION NUMBER: 21 (Cont'd)

TRANSACTION NAME: Add a CDP Number

DATA SOURCE: Course File Listing - Section A

FUNCTION: Add a new course and associated data to the course file

KEYIN REQUIRED: RETURN(EXEC) (From TRANS 21 page 1)

RESPONSE:

```
          ********** COURSE CDP # NNNN **********
01 CURRENT YEAR: CONVEN/LENGTH/CAP/BCAP/PLN INPUT/PLN B-INPUT
                 ---   /------  /----/ ---/------   /------
02 CUR YEAR  +1: CONVEN/LENGTH/CAP/BCAP/PLN INPUT/PLN B-INPUT
                 ---   /------  /---/---/------   /------
03 CUR YEAR  +2: CONVEN/LENGTH/CAP/BCAP/PLN INPUT/PLN B-INPUT
                 ---   /------  /---/---/------   /------
04 STUDENT CLASS CAPACITIES:
       MAX=---  MIN=----  PERS=---  EQUIP=---  SPACE=---
05 HISTORICAL CRS RATES:
   FAILURE=--.-%  NONAC DIS=--.-%  NOSHOW=--.-%  SETBACK=--.-%

KEY LINE NO CONTAINING DATA TO BE ENTERED OR CHANGED.
TO SKIP A DATA ITEM ON A LINE KEY RETURN; CURSOR WILL ADVANCE TO
THE NEXT DATA ITEM. TO EDIT REMAINING COURSE DATA, RETURN -
```

NOTES: 1) Typing errors may be corrected by backspacing over error and retyping.

2) Data is not stored until RETURN(EXEC) is keyed.

3) Dashes indicate the maximum number of characters which can be entered. It is not necessary to enter a character in each space.

4) Data for data items on this page may be entered or skipped as required. For example, to enter current year capacity data, key

1 RETURN(EXEC)

in response to the instruction at the bottom of the page. The cursor will be positioned under CONVEN. Key RETURN(EXEC). The cursor will be positioned under LENGTH. Again key RETURN(EXEC). The cursor will be positioned under CAP. Keyin capacity data, followed by RETURN(EXEC). The cursor will then move to BCAP. Keyin three (3) more RETURN(EXEC)s to reposition the cursor at the bottom of the page.

5) Historical course rates are shown on the Course file listing as decimal numbers, i.e., .030, which is 3%. To enter a 3% failure rate in line 5, keyin 0 3 . 0 %

TO TERMINATE PAGE: Key RETURN(EXEC).

RESPONSE: TRANS 21 page 3 (see next page).

MAINTENANCE TRANSACTION NUMBER: 21 (Cont'd)

TRANSACTION NAME: Add a CDP Number

DATA SOURCE: Course File Listing - Section A

FUNCTION: Add a new course and associated data to the course file

KEYIN REQUIRED: RETURN(EXEC) (from TRANS 21 page 2)

RESPONSE:

```
          ********** COURSE CDP # NNNN **********
06 ACT/DEPT=----  CIN=---------- CRS TITLE=------------------

07 COURSE UTILIZATION=---%  PLANNED AOB=-----  BACKLOG=----

08 DATE NEXT COURSE REVIEW=YY/MM/DD     STAFF UIC=----

09 CCMM=----------  PRIORITY CODE =--

10 COURSE TYPE=--  NEC AWARDED=----

11 OPTIONAL1=------  OPTIONAL2=------  OPTIONAL3=------
KEY LINE NO CONTAINING DATA TO BE ENTERED OR CHANGED.
TO SKIP A DATA ITEM ON A LINE KEY RETURN; CURSOR WILL ADVANCE TO
THE NEXT DATA ITEM. IF NO MORE EDITING REQ'D,KEY RETURN -
```

NOTES: 1) Typing errors may be corrected by backspacing over error and retyping.

2) Data is not stored until RETURN(EXEC) is keyed.

3) Dashes indicate the maximum number of characters which can be entered. It is not necessary to enter a character in each space.

4) Data for data items on this page may be entered or skipped as required.

TO TERMINATE PAGE: Key RETURN(EXEC).

RESPONSE: MENU.

MAINTENANCE TRANSACTION NUMBER: 23

TRANSACTION NAME: Add a Student/Instructor Ratio

DATA SOURCE: Course File Listing - Section B

FUNCTION: Add a new student/instructor ratio and associated data to a course that already exists in the Course file or to a course that is being added to the Course file during the current terminal session.

KEYIN REQUIRED: 23 RETURN(EXEC) (From MENU)

RESPONSE:

```
*ADD STUDENT/INSTRUCTOR RATIO DATA FOR THE FOLLOWING COURSE

CDP NUMBER ----

ACTIVITY/DEPT CODE ----
(IF SAME AS PREVIOUS TRANSACTION (NNNN),KEY RETURN.)
```

NOTES: 1) Typing errors may be corrected by backspacing over error and retyping.

2) Data is not stored until RETURN(EXEC) is keyed.

3) Dashes indicate the maximum number of characters which can be entered. It is not necessary to enter a character in each space.

4) If the activity/dept code to be entered is the same as the previous entry, it can be entered automatically by keying RETURN(EXEC). If the required activity/dept code is not the same as the code printed on the following line, the new code should be entered.

TO TERMINATE PAGE: Key RETURN(EXEC after entering Activity/Dept code.

RESPONSE: TRANS 23 page 2 (see next page).

MAINTENANCE TRANSACTION NUMBER: 23 (Cont'd)

TRANSACTION NAME: Add a Student/Instructor Ratio

DATA SOURCE: Course File Listing - Section B

FUNCTION: Add a new student/instructor ratio and associated data to a course that already exists in the Course file or to a course that is being added to the Course file during the current terminal session.

KEYIN REQUIRED: 23 RETURN(EXEC) (From MENU)

RESPONSE:

```
CDP# NNNN

   STUDENT/INSTRUCTOR RATIO  ---/1

   CONTACT HOURS AT THIS RATIO = ----

   RATIO TYPE = -
    1=LECTURE
    2=LAB
    3=COMB LEC/LAB


  TO ENTER ADDITIONAL RATIO DATA FOR THE ABOVE
  COURSE TYPE 1, ELSE KEY RETURN -
```

NOTES: 1) Typing errors may be corrected by backspacing over error and retyping.

2) Data is not stored until RETURN(EXEC) is keyed.

3) Dashes indicate the maximum number of characters which can be entered. It is not necessary to enter a character in each space.

4) All data items must be entered.

5) To add an additional student/instructor ratio for the same course, enter a 1 in response to the instruction at the bottom of the page.

TO TERMINATE PAGE: Key RETURN(EXEC).

RESPONSE: MENU

MAINTENANCE TRANSACTION NUMBER: 26

TRANSACTION NAME:	Add an Instructor Assignment
DATA SOURCE:	Course File Listing - Section C
FUNCTION:	Add a new instructor assignment and associated data to a course that already exists in the Course file or to a course that is being added to the Course file during the current terminal session.
KEYIN REQUIRED:	26 RETURN(EXEC) (From MENU)
RESPONSE:	

ADD INSTR ASSIGNMENT DATA FOR THE FOLLOWING INSTRUCTOR:

INSTRUCTOR ID(SSN) =---'--'---- IS ---% QUALIFIED

TO INSTRUCT IN THE FOLLOWING COURSE:

CDP NUMBER ----

ACTIVITY/DEPT CODE ----
(IF SAME AS PREVIOUS TRANSACTION (NNNN),KEY RETURN.)

ASSIGN FLAG = -
(ENTER 0 IF NOT TEACHING ABOVE COURSE; ELSE ENTER 1)

TO ENTER ADDITIONAL COURSE ASSIGNMENTS FOR THE ABOVE INSTRUCTOR TYPE 1, ELSE KEY RETURN -

NOTES: 1) Typing errors may be corrected by backspacing over error and retyping.

2) Data is not stored until RETURN(EXEC) is keyed.

3) Dashes indicate the maximum number of characters which can be entered. It is not necessary to enter a character in each space.

4) All data items must be entered.

5) If the activity/dept code to be entered is the same as the previous entry, it can be automatically entered by keying RETURN(EXEC).

6) To add an additional instructor to the same course, keyin 1 RETURN(EXEC) in response to the instruction at the bottom of the page.

TO TERMINATE PAGE: Key RETURN(EXEC).

RESPONSE: MENU

DOTS DATA BASE INSTRUCTOR FILE MAINTENANCE

The Instructor file contains rate/rank, report and rotate dates, NECS, and the activity profile data for each instructor sorted by ID number. As with the Course file, red-lined Instructor file listings will be the source document for entering change data on the WANG data cassette. A sample DOTS Instructor file listing is shown below:

DOTS INSTRUCTOR FILE LISTING (FILENAME = INST) PAGE : 45
ON 08/07/75 AT 1746

ID	RATE	IN	NAME	SCH	REP	ROT	PNEC	SNEC	NEC3	NEC4	NEC5
473626394	ARM1	DB	HANSEN	6E	7605	7906					
	CONT : 600	PREP : 100	FTRN : 0	FLET : 0	SKLV : 0						
	CDEV : 15	BKIN : 750	MDUT : 120	ADMN : 50	SLIB : 0						
	EQMT : 140	CTRN : 0	ISCH : 160	ANLV : 200							
480340003	HTC	CE	PUGH	6E	7404	7704		9568			
	CONT : 776	PREP : 48	FTRN : 0	FLET : 0	SKLV : 8						
	CDEV : 46	BKIN : 30	MDUT : 190	ADMN : 250	SLIB : 336						
	EQMT : 36	CTRN : 0	ISCH : 0	ANLV : 360							
481708667	HT2	R	JEFFERY	6E	7509	7809		9502			
	CONT : 500	PREP : 250	FTRN : 0	FLET : 100	SKLV : 0						
	CDEV : 50	BKIN : 20	MDUT : 100	ADMN : 50	SLIB : 30						
	EQMT : 100	CTRN : 60	ISCH : 200	ANLV : 620							
490462714	MMC	JM	EARNEST	6E	7509	7809	4294				
	CONT : 721	PREP : 125	FTRN : 0	FLET : 75	SKLV : 0						
	CDEV : 45	BKIN : 75	MDUT : 152	ADMN : 175	SLIB : 175						
	EQMT : 75	CTRN : 0	ISCH : 150	ANLV : 312							
492423681	ABM1	ML	STINSON	6E	7402	7702		9502			
	CONT : 204	PREP : 208	FTRN : 32	FLET : 100	SKLV : 0						
	CDEV : 0	BKIN : 100	MDUT : 852	ADMN : 100	SLIB : 0						
	EQMT : 104	CTRN : 0	ISCH : 240	ANLV : 240							
503622306	HT1	D	SMOUN	6E	7403	7703	4951	9502			
	CONT : 482	PREP : 200	FTRN : 66	FLET : 50	SKLV : 0						
	CDEV : 36	BKIN : 36	MDUT : 140	ADMN : 0	SLIB : 30						
	EQMT : 60	CTRN : 120	ISCH : 200	ANLV : 580							

MAINTENANCE TRANSACTION NUMBER: 3

TRANSACTION NAME: Change Instructor Data

DATA SOURCE: Instructor File Listing

FUNCTION: Change the data contained in any data field listed in the Instructor file listing (except the ID number)

KEYIN REQUIRED: 3 RETURN(EXEC) (From MENU)

RESPONSE:

CHANGE INSTRUCTOR DATA FOR THE FOLLOWING INSTRUCTOR:

INSTRUCTOR ID(SSN) =---'--'----

ACTIVITY/DEPT CODE ----
(IF SAME AS PREVIOUS TRANSACTION (NNNN),KEY RETURN.)

NOTES: 1) Typing errors may be corrected by backspacing over error and retyping.

2) Data is not stored until RETURN(EXEC) is keyed.

3) Dashes indicate the maximum number of characters which can be entered. It is not necessary to enter a character in each space.

4) If the activity/dept code to be entered is the same as the previous entry, it can be entered automatically by keying RETURN(EXEC).

TO TERMINATE PAGE: Key RETURN(EXEC) after entering activity/dept code.

RESPONSE: TRANS 3 page 2 (see next page).

PRECEDING PAGE BLANK-NOT FILMED

MAINTENANCE TRANSACTION NUMBER: 3 (Cont'd)

TRANSACTION NAME: Change Instructor Data

DATA SOURCE: Instructor File Listing

FUNCTION: Change the data contained in any data field listed in the Instructor file listing (except the ID number)

KEYIN REQUIRED: RETURN(EXEC) (From TRANS 3 page 1)

RESPONSE:

```
   ID = XXX-XX-XXXX     NAME = XXXXXXXXXXXXXXXX

01 NAME ----------------  INITIALS --

02 RATE -----             ACT/DEPT = ----

03 REPORT DATE = YY/MM/DD   ROTATE DATE = YY/MM/DD

04 PRIMARY NEC = ----  SECONDARY NEC = ----

05 ADDITIONAL NECS = ----   ----   ----

KEY LINE NO CONTAINING DATA TO BE ENTERED OR CHANGED.
TO SKIP A DATA ITEM ON A LINE KEY RETURN; CURSOR WILL ADVANCE TO
THE NEXT DATA ITEM. TO EDIT REMAINING INST DATA, RETURN -
```

NOTES: 1) Typing errors may be corrected by backspacing over error and retyping.

2) Data is not stored until RETURN(EXEC) is keyed.

3) Dashes indicate the maximum number of characters which can be entered. It is not necessary to enter a character in each space.

4) Data items may be entered or omitted as required. For example, to enter only a new rotate date, key 3 RETURN(EXEC) in response to the instruction at the bottom of the page. The cursor will advance to the first item on line 3 (report date). Key RETURN(EXEC). The cursor will advance to the next item on line 3 (ROTATE DATE). Key in the new rotate date followed by RETURN(EXEC). The cursor will return to the bottom of the page.

TO TERMINATE PAGE: RETURN(EXEC).

RESPONSE: TRANS 3 page 3 (see next page).

MAINTENANCE TRANSACTION NUMBER: 3 (Cont'd)

TRANSACTION NAME: Change Instructor Data

DATA SOURCE: Instructor File Listing

FUNCTION: Change the data contained in any data field listed in the Instructor file listing (except the ID number)

KEYIN REQUIRED: RETURN(EXEC) (From TRANS 3 page 2)

RESPONSE:

```
   ID = XXX-XXX-XXXX     NAME = XXXXXXXXXXXXXXXX

 **NOTE: ALL INSTRUCTOR ACTIVITY DATA ARE HOURS/YEAR

06 STUDENT CONTACT = ----          CURRICULUM DEVELOPMENT = ----
07 CLASS PREP/RELATED DUT = ----   INST TRNG(BREAK-IN) = ----
08 EQUIP-TRNG/MAINT = ----         CROSS TRNG(LOW STUD LOAD)=----
09 FORMAL INST SCH = ----          FACTORY TRNG/WKSHOPS = ----
10 SPEC FLEET ASSISTANCE= ----     MIL DUT/PHYS TRNG/OTHER = ----
11 SUPERVISION/ADMIN = ----        ANNUAL LEAVE = ----
12 SICK LEAVE = ----               SPECIAL LIBERTY = ----

KEY LINE NO CONTAINING DATA TO BE ENTERED OR CHANGED.
TO SKIP A DATA ITEM ON A LINE KEY RETURN; CURSOR WILL ADVANCE TO
THE NEXT DATA ITEM. IF NO MORE EDITING REQUIRED,KEY RETURN -
```

NOTES: 1) Typing errors may be corrected by backspacing over error and retyping.

2) Data is not stored until RETURN(EXEC) is keyed.

3) Dashes indicate the maximum number of characters which can be entered. It is not necessary to enter a character in each space.

4) Only those activity hours which have changed need be entered. Key line number containing activity category to be changed. To enter new hours for only one category on a line, key RETURN(EXEC) when the cursor is positioned to the category for which no change is required.

TO TERMINATE PAGE: Key RETURN(EXEC).

RESPONSE: MENU

MAINTENANCE TRANSACTION NUMBER: 13

TRANSACTION NAME: Delete Instructor ID

DATA SOURCE: Instructor File Listing; Course File Listing - Section C

FUNCTION: Deletes the ID number and all associated data from the Instructor file and deletes all course assignments for the ID number from the Course file

KEYIN REQUIRED: 13 RETURN(EXEC) (From MENU)

RESPONSE:

```
* DELETE THE FOLLOWING INSTRUCTOR

INSTRUCTOR ID(SSN) = ___-__-____

ACTIVITY/DEPT CODE ----
(IF SAME AS PREVIOUS TRANSACTION (NNNN),KEY RETURN.)
```

NOTES: 1) Typing errors may be corrected by backspacing over error and retyping.

2) Data is not stored until RETURN(EXEC) is keyed.

3) Dashes indicate the maximum number of characters which can be entered. It is not necessary to enter a character in each space.

4) If the activity/dept code to be entered is the same as the previous entry, it can be entered automatically by keying RETURN(EXEC).

TO TERMINATE PAGE: Key RETURN(EXEC) after entering activity/dept code.

RESPONSE: MENU

MAINTENANCE TRANSACTION NUMBER: 18

TRANSACTION NAME: Change Instructor ID

DATA SOURCE: Instructor File Listing; Course File Listing - Section C

FUNCTION: Change only the ID number from the value currently listed to a new value

KEYIN REQUIRED: 18 RETURN(EXEC) (From MENU)

RESPONSE:

```
   * CHANGE THE FOLLOWING INSTRUCTOR ID(SSN)

OLD INSTRUCTOR ID(SSN) =  ---¯--¯----

NEW INSTRUCTOR ID(SSN) =  ---¯--¯----

      ACTIVITY/DEPT CODE ----
      (IF SAME AS PREVIOUS TRANSACTION (NNNN),KEY RETURN.)
```

NOTES: 1) Typing errors may be corrected by backspacing over error and retyping.

2) Data is not stored until RETURN(EXEC) is keyed.

3) Dashes indicate the maximum number of characters which can be entered. It is not necessary to enter a character in each space.

4) If the activity/dept code to be entered is the same as the previous entry, it can be entered automatically by keying RETURN(EXEC).

5) All additional instructor data changes associated with the instructor whose ID number has been changed should refer to the new instructor ID number.

6) This transaction changes the ID number in both the Instructor file and in the Course file (instructor assignments - Section C of Course file listing).

TO TERMINATE PAGE: Key RETURN(EXEC) after entry of activity/dept code.

RESPONSE: MENU

MAINTENANCE TRANSACTION NUMBER: 24

TRANSACTION NAME: Add Instructor ID

DATA SOURCE: Instructor File Listing

FUNCTION: Add a new instructor ID and all associated data to the Instructor file.

KEYIN REQUIRED: 24 RETURN(EXEC) (From MENU)

RESPONSE:

```
* ADD THE FOLLOWING INSTRUCTOR

INSTRUCTOR ID(SSN) =---'--'----

LAST NAME ----------------
FIRST,MIDDLE INITIALS --

RATE/RANK = -----

ASSIGNED TO
ACTIVITY/DEPT CODE ----
(IF SAME AS PREVIOUS TRANSACTION (NNNN),KEY RETURN.)
```

NOTES: 1) Typing errors may be corrected by backspacing over error and retyping.

2) Data is not stored until RETURN(EXEC) is keyed.

3) Dashes indicate the maximum number of characters which can be entered. It is not necessary to enter a character in each space.

4) All requested data must be entered.

5) If the activity/dept code to be entered is the same as the previous entry, it can be entered automatically by keying RETURN(EXEC).

TO TERMINATE PAGE: Key RETURN(EXEC) after entering activity/dept code.

RESPONSE: TRANS 24 page 2 (see next page).

MAINTENANCE TRANSACTION NUMBER: 24 (Cont'd)

TRANSACTION NAME: Add Instructor ID

DATA SOURCE: Instructor File Listing

FUNCTION: Add a new instructor ID and all associated data to the Instructor file

KEYIN REQUIRED: RETURN(EXEC) (From TRANS 24 page 1)

RESPONSE:

```
     ID = XXX-XX-XXXX     NAME = XXXXXXXXXXXXXXXX

01   NAME ----------------  INITIALS --

02   RATE -----             ACT/DEPT = ----

03 REPORT DATE = YY/MM/DD   ROTATE DATE = YY/MM/DD

04 PRIMARY NEC = -----  SECONDARY NEC = -----

05 ADDITIONAL NECS = ----   ----   ----

KEY LINE NO CONTAINING DATA TO BE ENTERED OR CHANGED.
TO SKIP A DATA ITEM ON A LINE KEY RETURN; CURSOR WILL ADVANCE TO
THE NEXT DATA ITEM. TO EDIT REMAINING INST DATA, RETURN -
```

NOTES: 1) Typing errors may be corrected by backspacing over error and retyping.

2) Data is not stored until RETURN(EXEC) is keyed.

3) Dashes indicate the maximum number of characters which can be entered. It is not necessary to enter a character in each space.

4) Data items on lines 1 and 2 should not be entered unless errors were made when these data items were entered on the previous page.

5) Data items on lines 3-5 may be entered or skipped as required. A data item can be skipped by keying RETURN(EXEC), the cursor will advance to the next data item on the line or return to the instruction at the bottom of the page.

TO TERMINATE PAGE: Key RETURN(EXEC).

RESPONSE: TRANS 24 page 3 (see next page).

MAINTENANCE TRANSACTION NUMBER: 24 (Cont'd)

TRANSACTION NAME: Add Instructor ID

DATA SOURCE: Instructor File Listing

FUNCTION: Add a new instructor ID and all associated data to the Instructor file

KEYIN REQUIRED: RETURN(EXEC) (From TRANS 24 page 2)

RESPONSE:

```
   ID = XXX-XXX-XXXX     NAME = XXXXXXXXXXXXXXXX

 **NOTE: ALL INSTRUCTOR ACTIVITY DATA ARE HOURS/YEAR

06 STUDENT CONTACT = ----          CURRICULUM DEVELOPMENT = ----
07 CLASS PREP/RELATED DUT = ----   INST TRNG(BREAK-IN) = ----
08 EQUIP-TRNG/MAINT = ----         CROSS TRNG(LOW STUD LOAD)=----
09 FORMAL INST SCH = ----          FACTORY TRNG/WKSHOPS = ----
10 SPEC FLEET ASSISTANCE= ----     MIL DUT/PHYS TRNG/OTHER = ----
11 SUPERVISION/ADMIN = ----        ANNUAL LEAVE = ----
12 SICK LEAVE = ----               SPECIAL LIBERTY = ----

KEY LINE NO CONTAINING DATA TO BE ENTERED OR CHANGED.
TO SKIP A DATA ITEM ON A LINE KEY RETURN; CURSOR WILL ADVANCE TO
THE NEXT DATA ITEM. IF NO MORE EDITING REQUIRED,KEY RETURN -
```

NOTES: 1) Typing errors may be corrected by backspacing over error and retyping.

2) Data is not stored until RETURN(EXEC) is keyed.

3) Dashes indicate the maximum number of characters which can be entered. It is not necessary to enter a character in each space.

4) Entry of 0 hours is not required. Omitted activity categories will automatically default to 0 hours.

5) Data for activity categories can be entered or omitted as required. To omit a data item key RETURN(EXEC); the cursor will advance to the next data item or will return to the instruction at the bottom of the page.

TO TERMINATE PAGE: Key RETURN(EXEC).

RESPONSE: MENU

DOTS DATA BASE FACILITY FILE MAINTENANCE

The Facility file contains building/room assignments, including space capacity, required and available hours, sorted by course CDP number. Red-lined Facility file listings will be the source document for entering Facility file changes on the WANG data cassette. A sample DOTS Facility file listing is shown below:

DOTS FACILITIES FILE LISTING (FILENAME = FACIL)
ON 08/07/7- AT 1746

CDP	SCH	BLDG	RM	RMCAP	RMTP	REQHRS	AVHRS
1130	430	67A	116	15	3	7.0	1750
1130	430	67A	118	15	2	30.0	1750
1130	430	67A	121	10	2	20.0	1750
1130	430	67A	137	15	2	57.0	1750
1130	430	67A	234	15	1	138.0	1750
1262	430	67A	121	0	2	10.0	1750
1262	430	67A	221	0	1	46.0	1750
205K	430	67A	116	10	2	16.0	1750
205K	430	67A	221	20	1	19.0	1750
2625	430	67A	112	25	1	12.0	1750
2625	430	67A	118	6	2	23.0	1750
323G	430	67A	149	20	2	48.0	1750
323G	430	67A	209	40	1	148.0	1750
506T	430	67A	116	8	2	9.0	1750
506T	430	67A	221	30	1	19.0	1750
506V	430	67A	116	12	2	29.0	1750
506V	430	67A	221	30	1	20.0	1750
506Y	430	67A	221	20	1	8.0	1750
506Z	430	67A	112	25	1	10.0	1750
506Z	430	67A	134	16	4	11.0	625
507C	430	67A	118	18	1	13.0	1750
507C	430	67A	134	16	4	22.0	625
507F	430	67A	118	16	3	21.0	1750
507F	430	67A	229	40	1	14.0	1750
540Z	430	67A	221	20	1	20.5	1750
6025	430	67A	108	20	3	15.0	1750

MAINTENANCE TRANSACTION NUMBER: 2

TRANSACTION NAME: Change Building/Room Data

DATA SOURCE: Facility File Listing

FUNCTION: Change the value of any data item (except building/room number) associated with a building/room number course assignment as identified in the Facility file listing.

KEYIN REQUIRED: 2 RETURN(EXEC) (From MENU)

RESPONSE:

```
*CHANGE FACILITY DATA FOR THE FOLLOWING COURSE

CDP NUMBER ----

ACTIVITY/DEPT CODE ----
(IF SAME AS PREVIOUS TRANSACTION (NNNN),KEY RETURN.)

BUILDING/ROOM NUMBER = ----/----
REVISED DATA:
ROOM CAPACITY(#STDT)=---    ROOM TYPE = -
                          (1=LEC,2=LAB,3=BOTH,4=TRNER)

HOURS REQ'D PER XXXX CONV =----  TOT HR AVAIL/YEAR=-----
TO ENTER ADDITIONAL FACILITY CHANGES FOR THE ABOVE
COURSE TYPE 1, ELSE KEY RETURN -
```

NOTES: 1) Typing errors may be corrected by backspacing over error and retyping.

2) Data is not stored until RETURN(EXEC) is keyed.

3) Dashes indicate the maximum number of characters which can be entered. It is not necessary to enter a character in each space.

4) If the activity/dept code to be entered is the same as the previous entry, it can be entered automatically by keying RETURN(EXEC).

5) RETURN(EXEC) must be keyed after building number to position cursor for room number entry.

6) Only building/room number must be entered. Remaining data items may be entered or skipped as required. To skip a data item, key RETURN(EXEC) when the cursor is positioned at the data item to be skipped.

7) To enter changes for other facilities associated with the same course, keyin 1 RETURN(EXEC) in response to the instruction at the bottom of the page.

TO TERMINATE PAGE: Key RETURN(EXEC)

RESPONSE: MENU

MAINTENANCE TRANSACTION NUMBER: 14

TRANSACTION NAME: Delete Building/Room Number

DATA SOURCE: Facility File Listing

FUNCTION: Delete a building/room assignment and all associated data for the specified course CDP number

KEYIN REQUIRED: 14 RETURN(EXEC) (from MENU)

RESPONSE:

```
*DELETE FACILITY DATA FOR THE FOLLOWING COURSE

 CDP NUMBER ----

 ACTIVITY/DEPT CODD ----
 (IF SAME AS PREVIOUS TRANSACTION (NNNN),KEY RETURN.)


 BUILDING/ROOM NUMBER = ----/----
```

NOTES: 1) Typing errors may be corrected by backspacing over error and retyping.

2) Data is not stored until RETURN(EXEC) is keyed.

3) Dashes indicate the maximum number of characters which can be entered. It is not necessary to enter a character in each space.

4) If the activity/dept code to be entered is the same as the previous entry, it can be entered automatically by keying RETURN(EXEC).

5) RETURN(EXEC) must be keyed after building number to position cursor for room number entry.

TO TERMINATE PAGE: Key RETURN(EXEC) after entering building/room number.

RESPONSE: MENU

MAINTENANCE TRANSACTION NUMBER: 19

TRANSACTION NAME: Change Building/Room Number

DATA SOURCE: Facility File Listing

FUNCTION: Change only the building number and/or the room number.

KEYIN REQUIRED: 19 RETURN(EXEC) (From MENU)

RESPONSE:

```
*CHANGE BUILDING/ROOM NUMBER DATA FOR THE FOLLOWING COURSE

 CDP NUMBER ----

 ACTIVITY/DEPT CODE ----
 (IF SAME AS PREVIOUS TRANSACTION (NNNN),KEY RETURN.)

OLD BUILDING/ROOM NUMBER = ----/----

NEW BUILDING/ROOM NUMBER = ----/----
```

NOTES: 1) Typing errors may be corrected by backspacing over error and retyping.

2) Data is not stored until RETURN(EXEC) is keyed.

3) Dashes indicate the maximum number of characters which can be entered. It is not necessary to enter a character in each space.

4) If the activity/dept code to be entered is the same as the previous entry, it can be entered automatically by keying RETURN(EXEC).

5) RETURN(EXEC) must be keyed after building number to position cursor for room number entry.

6) This transaction changes all occurrences of the old building/ room number in the Facility file to the specified new number.

TO TERMINATE PAGE: Key RETURN(EXEC) after entering building/room number.

RESPONSE: MENU

MAINTENANCE TRANSACTION NUMBER: 25

TRANSACTION NAME: Add a Building/Room Number

DATA SOURCE: Facility File Listing

FUNCTION: Add a facility identified by building/room number and all associated data to the Facility file

KEYIN REQUIRED: 25 RETURN(EXEC) (From MENU)

RESPONSE:

```
*ADD FACILITY DATA FOR THE FOLLOWING COURSE:

CDP NUMBER ----

ACTIVITY/DEPT CODE ----
(IF SAME AS PREVIOUS TRANSACTION (NNNN),KEY RETURN.)

BUILDING/ROOM NUMBER = ----/----

ROOM CAPACITY(#STDT)=---   ROOM TYPE = -
                          (1=LEC,2=LAB,3=BOTH,4=TRNER)

HOURS REQ'D PER XXXX CONV = ----  TOT HR AVAIL/YEAR=-----
TO ENTER ADDITIONAL FACILITY DATA FOR THE ABOVE
COURSE TYPE 1, ELSE KEY RETURN -
```

NOTES: 1) Typing errors may be corrected by backspacing over error and retyping.

2) Data is not stored until RETURN(EXEC) is keyed.

3) Dashes indicate the maximum number of characters which can be entered. It is not necessary to enter a character in each space.

4) If the activity/dept code to be entered is the same as the previous entry, it can be entered automatically by keying RETURN(EXEC).

5) RETURN(EXEC) must be keyed after building number to position cursor for room number entry.

6) Data must be entered for all data items.

7) If more than one facility is to be identified for the same course, keyin 1 RETURN(EXEC) in response to the instruction at the bottom of the page.

TO TERMINATE PAGE: Key RETURN(EXEC)

RESPONSE: MENU

DATA BASE MAINTENANCE OPERATIONAL PROCEDURES - CASSETTE TRANSMISSION TO NCSS

Procedures to transmit WANG cassette tapes containing data base maintenance transactions to the NCSS timesharing environment are presented in this section. These procedures are designed to be run on the WANG computer system located at COMTRAPAC. This system consists of a 2200 System CPU (16K memory required), Option 2 (General I/O ROM), a telecommunications modem, a CRT display and keyboard, a single cassette tape drive, a high-speed printer, and a disk storage unit.

Upon receipt of a maintenance data cassette tape from a training activity, the Data Base Manager (DBM) at COMTRAPAC must transfer the data from the cassette tape to the WANG disk, sort the data by transaction type, and then transmit the data via telecommunication from the WANG disk to the NCSS disk. The two WANG programs required to accomplish these tasks are DBTAPERD and TC-NCSS. The following procedures assume that the DBM is proficient in the operation of the WANG System in the following areas: loading and running programs from either tape or disk, editing programs in core, cataloging data files on the disk, and running the disk sort program. These procedures are also documented in WANG reference manuals.

TRANSFER DATA FROM TAPE TO DISK. The program DBTAPERD is required to transfer data from the cassette tape to the disk. DBTAPERD must first be loaded and run. A menu of special function (S.F.) options will be displayed. S.F. 1 will generate an unformatted listing of the transaction tape and will write the tape records to the disk file "DB-INPUT." Prior to keying S.F.1 the disk file "DB-INPUT" must be cataloged on the R-disk. The number of disk sectors required for "DB-INPUT" is a function of the number of maintenance transactions to be written in data file "DB-INPUT." Each transaction requires one disk sector. A full cassette tape contains approximately 650 transactions. Up to two full tapes (or 1300 transactions) may be read into "DB-INPUT" prior to executing the sort program. Files larger than 1300 transactions require excessive time to edit and process in the NCSS environment.

After "DB-INPUT" has been cataloged on the R-disk, key S.F.1. The program will respond:

STOP; MOUNT TRANSACTION TAPE CASSETTE, THEN KEY CONTINUE

The data transfer is automatic after the tape has been mounted and CONTINUE has been keyed. A hard-copy listing of the tape will be printed as each record is read into "DB-INPUT." To append additional transactions to "DB-INPUT," mount another tape and key S.F.1 again. The data from the second tape will then be appended to the data from the first tape in "DB-INPUT."

SORT "DB-INPUT." After the desired number of maintenance transactions have been read into "DB-INPUT," the disk file "DB-INPUT" should be sorted using

the WANG utility disk sort program. The format of the records in "DB-INPUT" is shown below:

FIELD NUMBER	NUMBER OF BYTES	CONTENTS
1	2	Transaction Type
2	4	Activity/dept Code
3	4	CDP Number
4	40	Transaction Record Bytes 11-50

"DB-INPUT" must be sorted by transaction type (field 1). Secondary sorts may be made on activity/dept code and/or CDP number at the discretion of the DBM. The output file for the utility sort routine should be named "DB-SORT2."

LIST SORTED TRANSACTION FILE. To generate a sequence-numbered and formatted listing of the sorted transaction file "DB-SORT2" load and run "DBTAPERD." Then key S.F.2. A hard-copy listing containing the sequence number of each record in the DB-SORT2" file, a field heading for each field in the transaction record and a statement noting the transaction function will be printed with no further user intervention.

This formatted listing should be reviewed by the DBM according to the general maintenance procedures discussed earlier in Section II. Upon completion of the NCSS transaction processing, errors detected by RAMIS should be noted on this listing. The annotated copy of the listing should then be returned to the appropriate training activity. This will inform each activity which transactions have been included in the RAMIS data base and which transactions have been rejected. Rejected transactions can then be corrected and resubmitted by the activity during the next maintenance cycle.

TRANSMIT TRANSACTION FILE TO NCSS ENVIRONMENT. The WANG program required to transmit the transaction file to NCSS is TC-NCSS. First, TC-NCSS must be loaded. Prior to running TC-NCSS, statements 710 and 715 should be examined and modified as required.

Statement 710 should specify the name of the disk file containing the sorted transaction file. Statement 715 should be deleted. TC-NCSS should then be run. The following message will be displayed:

```
ENTER END OF LINE SEQUENCE IN HEX, EG.
 CR/LF/7F   =  0D 0A 7F
 CR/LF/00   =  0D 0A 00
 LF/CR/7F   =  0A 0D 7F
```

The user should respond:

```
0A 0D 7F
```

The following message will then be displayed:

```
WANG 2200-B  TELECOMMUNICATIONS

  KEYBOARD INTERACTION WITH T.C.  AUTO-OUTPUT TO LINE PRINTER

REQUIRES 2200-B (8K)  CRT/KEYBOARD AND 2227 BOARD
     WITH OPTION 2 I/O ROM
S.F.'12 CRT/PRINTER OUTPUT
S.F.'13 CRT OUTPUT
S.F.'14 EJECT PRINTER PAGE
S.F.'15 DISPLAY OF S.F.POSSIBILITIES
STOP: KEY APPROPRIATE SPECIAL FUNCTION
```

The user should respond by keying S.F.12.

Prior to dialing the local NCSS telephone number the user should verify the following:

1) Printer should be on and selected.

2) The pushbuttons on the 2227 Telecommunications Controller should be set as follows:

BUTTON	POSITION
LSB	DOWN
OPS	UP
PAR	DOWN
NB2	UP
NB1	DOWN

3) The 300 baud transmission rate button on the 2227 Telecommunications Controller should be depressed.

After verifying the above switch positions, the user should dial the local NCSS telephone number. When a continuous tone is heard the telephone should be placed in the modem. The user should then keyin the speed identification characters: o (letter). When the NCSS system responds:

```
CSS ONLINE
```

the user should follow the system login procedures outlined in Section 2 of the NCSS VP/CSS Reference Manual, form 106-4. When the login procedure is complete, the system will respond:

```
CSS.300 06/15/76
hh.mm.ss
```

At this point, the user can initiate the file transmission process by keying:

```
EDIT FILENAME FILETYPE
```

where FILENAME and FILETYPE may be any eight character designator. A recommended convention is to assign the training activity name to the FILENAME and the current data to the FILETYPE, i.e., ASW 0910. The system will respond:

NEW FILE
NONSTANDARD FILETYPE
INPUT

At this point, the user should depress the reset button on the keyboard. The system will respond:

READY:

The user should now keyin S.F.10. At this point the transactions stored in the disk file named in statement 710 will be printed on the printer as they are read into the NCSS file named in the EDIT statement. If no transmission error occurs, all transactions will be read from the WANG disk file to the NCSS file with no further intervention. After the last transaction has been printed and sent to NCSS, printing will stop. The user should then key:

RETURN(EXEC)

The system will respond:

EDIT

The user should key:

LINENO

The system will respond with the line number of the last line entered. If no errors have occurred, this line number should be equal to the number of transactions transmitted. To perform an additional check, the user should key:

PRINT

This will cause NCSS to print the last transaction entered. This should be identical to the last transaction listed on the numbered, formatted listing of the sorted transaction file. To store the new NCSS file, the user should then key:

FILE

The system will respond with:

hh.mm.ss (time-of-day)

To recover from a transmission error, which may be difficult to detect but may manifest itself by halting transmission, thereby halting record

printing at the printer, the steps outlined below should be followed.

First, depress the reset button, followed by keying

RETURN(EXEC)

if the system responds

EDIT

then follow the steps outlined above to determine the last record transmitted. If the system responds

VP

the user must first key

BACK RETURN(EXEC)

If the system then responds EDIT, follow the steps above. If no response is given, key an additional

RETURN(EXEC)

The system should then respond EDIT.

To resume transmission at the transaction following the last transaction sent, key:

SAVE

The system will respond

INPUT:

The user should then depress the reset button, and insert statement 715 into the NCSS program. Statement 715 should be constructed as follows:

715 DSKIP NNN S

where NNN is the last transaction number transmitted. After inserting statement 715, the user should key S.F.10. Transmission will continue beginning with the transaction following the last transaction sent.

DATA BASE MAINTENANCE OPERATIONAL PROCEDURES - PROCESSING AT THE NCSS COMPUTER

Figure I-1 depicts the general flow of maintenance items in the DOTS system. In Table II-4 which follows, all types of maintenance activity that can be directed at the three files of the DOTS system are itemized.

All maintenance transactions originate at the WANG computer. For a given maintenance run, all transactions are accumulated onto a single transaction tape. Each transaction is identified by a format number; the transactions appear on the tape in sorted order by this format number. A specific maintenance run may be made with one or an indefinite number of transactions of any one or all formats.

Once the tape has been transferred to the NCSS system, the transactions reside in a specific NCSS file identified by a filename and filetype. To initiate the action which will ultimately result in presentation of the transaction data to the RAMIS system for execution, two executive sequences have been provided. The action of each is indicated by the following description. Note that for each, the filename/filetype of the input transaction file are provided as execution arguments.

To Initiate Action, Type	Results
super fn ft (CR) *	(1) Existing maintenance utility files are erased. (2) Input transactions from the file specified as fn ft are processed by the Transaction Processing Program. If errors are detected, the user is so notified. (3) Output utility files are created as needed. The output files and their relation to input transactions are indicated in Table II-4. If one or more transaction formats are missing in the input, one or more output files will not be established. (4) If explicit errors have occurred during the run of the Transaction Processing Program, the user is so notified and further processing aborted at this point. If no errors were detected, operation proceeds with Step (5) below.

* (CR) = Carriage Return or RETURN(EXEC) or CR/LF.

Format	Description	Output Name	File Type
2	Change a Facility file level 2 non-key field	DSC	RAM
3	Change an Instructor file level 1 non-key field	DSC	RAM
6	Change a Course file level 1 non-key field	DSC	RAM
7	Change a Course file student/instructor ratio (SIR) segment non-key field	DSC	RAM
8	Change a Course file instructor segment non-key field	DSC	RAM
11	Delete a course CDP number	DSA	RAM
12	Delete a student/instructor ratio (SIR) segment from the Course file	DSA	RAM
13	Delete an instructor ID	M13EXA	EXEC
14	Delete a building/room segment from the Facilities file	DSA	RAM
15	Delete a school code from all files	M15EXA	EXEC
16	Change a course CDP number	M16EXA	EXEC
17	Change a student/instructor ratio in the Course file	M17EXA	EXEC
18	Change an instructor ID	M18EXA	EXEC
19	Change a building/room number in the Facilities file	M19EXA	EXEC
20	Change a school code in all files	M20EXA	EXEC
21	Add a new course CDP to the Course and Facilities files	DSB	RAM

DATA BASE MAINTENANCE TRANSACTIONS
AND MAINTENANCE TRANSACTION FILES

TABLE II-4

Format	Description	Output Name	File Type
23	Add a new student/instructor ratio (SIR) segment to the Course file	DSB	RAM
24	Add a new instructor segment to the Course file	DSB	RAM
25	Add a new building/room segment to the Facilities file	DSB	RAM
26	Add a new instructor segment to an existing course	DSB	RAM
27	Delete an instructor segment from an existing course	DSB	RAM

TABLE II-4 (Cont'd)

To Initiate Action, Type	Results
	(5) Table II-5 indicates the order of processing for the maintenance utility files. This order must be maintained to assure that maintenance activity is scheduled as: all deletions, all changes, all additions. Within the NCSS executive sequence, all files are processed in the order shown in Table II-5 via a two-step logic. First, the file's existence is tested; if it is absent from the system, processing continues with the next utility file in line. If present, the data residing in the utility file is presented to the RAMIS system to accomplish the required maintenance.
superx fn ft (CR)	(1), (2), and (3) - As above, with executive = SUPER.
	(4) The user is notified of any errors detected during the run of the Transaction Processing Program and operation ceases at this point. Any required maintenance utility files will have been created.

After executive = SUPERX has completed, the user may cause further processing of the maintenance utility files by the following sequence of operations:

listf fn ft (CR) — Where fn ft is the name and type of the maintenance file of concern. This is an invocation of the standard NCSS LISTF function; response from the NCSS system will be the usual message indicating the number of items in the file or a message indicating that the file does not exist.

Once the given file's existence has been confirmed, the file's contents may be presented to the RAMIS system for processing by the keyboard sequences shown in Table II-5. For instance, to process DSB/RAM, enter the following:

ram dsb ram (CR)

The relative sequence of the file processing as shown in Table II-2 is critical. Processing of multiple files in an order differing from that shown will result in confusing and unpredictable errors.

Order of Execution	Utility Maintenance File Filename/Filetype		To Initiate RAMIS Processing, Enter:
1	DSA	RAM	ram dsa ram (CR)
2	M13EXA	EXEC	m13exa (CR)
3	M15EXA	EXEC	m15exa (CR)
4	M16EXA	EXEC	m16exa (CR)
5	M17EXA	EXEC	m17exa (CR)
6	M18EXA	EXEC	m18exa (CR)
7	M19EXA	EXEC	m19exa (CR)
8	M20EXA	EXEC	m20exa (CR)
9	DSB	RAM	ram dsb ram (CR)
10	DSC	RAM	ram dsc ram (CR)

UTILITY MAINTENANCE FILE
RAMIS PROCESSING

TABLE II-5

DATA BASE MAINTENANCE OPERATIONAL PROCEDURES -

GENERAL PURPOSE UTILITY PROGRAMS

Three general purpose executive sequences have been developed to assist the user with common data base maintenance tasks. These executives bear the names of the files with which they are associated, i.e., FACIL, INST, and COURSE. As with all NCSS executives, the sequence is initiated by typing in the executive name. These three executives require one (and only one) execution-time arguments. The following indicates the selections:

Executive Name	Arguments
FACIL	DUMP
INST	RELOAD
COURSE	FNS
	INDEX
	DESC
	DESCLOAD

As stated above, the execution arguments are mutually exclusive and must be invoked one at a time. Selection of the executive by the same name as the RAMIS file concerned is self-explanatory. The arguments elicit the following actions:

DUMP - A RAMIS dump function is executed against the selected file. In the case of the COURSE and INST files, auxiliary storage is used due to the size of the files. When completed, the file dump will be reformatted (not required for the FACIL file) and the resulting records will be stored in on-line disk storage. A subsequent invocation of the same executive with the RELOAD option (explained below) uses this reformatted dump file to erase the file concerned and effect its complete reload. The reformatted dump records reside in the following files (filename/filetype):

Facilities File - FACIL/DUMP

Instructor File - INST/DUMP2

Course File - COURSE/DUMP2

RELOAD - Using the dump file created by the DUMP option as explained above, this argument forces the erase and complete reload of the file concerned. After successful execution, the RAMIS file will be in the exact state as existed at the time that the DUMP was made.

FNS - This option causes an on-line listing of the executive sequences associated with these three general purpose executives; e.g., the following entry:

course fns CR

will cause the listing of executives E7, E8, and E9. The FNS option is really only of use to those concerned with system or program maintenance.

INDEX - This argument causes a RAMIS listing of the selected file's internal RAMIS description.

DESC - This option lists the selected file's RAMIS description as it exists in card-image form. For each of the three files, the applicable description is stored in the NCSS system as follows (filename/filetype):

Facilities File - FACIL/DESC

Instructor File - INST/DESC

Course File - COURSE/DESC

DESCLOAD - This entry causes the applicable file's description to be entered into the RAMIS system. This actfon completely erases the previous file description as well as the data file itself. This erase action is complete and the previous description and file contents will be irretrievably lost. If the file description (found in filename/DESC) has been modified before DESCLOAD, then a subsequent attempt at RELOAD will fail unless the related RAMIS functions associated with RELOAD have been modified in conjunction with the description change.

The following three executives utilize the reformatted dump files FACIL/DUMP, INST/DUMP2, and COURSE/DUMP2 to produce a listing of the current file contents. All fields in all records are listed. No arguments are required in the operation of these executives. At the time of this document, all printer listings were being routed to the San Diego office of National CSS. The user may pick up the printer output at this location.

DOTS File	Printer DUMP Executive Name
Facilities	PFACIL
Instructor	PINST
Course	PCOURSE

To assist the user in effective data base maintenance, a set of forms can be produced via printer. These forms can be used in a fill-in-the-blanks manner

to note the requirement for data base changes. No arguments are required in the invocation of these executives. No forms are provided for the Facilities file.

DOTS File	Blank Forms Executive Name
Instructor	FINST
Course	FCOURSE

SECTION III

TRAINING PROCESS FLOW MODEL

MODEL DESCRIPTION AND USER'S GUIDE

TPF MODEL ARCHITECTURE

The objectives of the Training Process Flow model are:

a. To provide training management with an ability to assess the effects upon the training system of projected changes in:

- o Demand
- o Scheduling
- o Capacities

b. To develop model output data formats that provide high informational content to training management, so that effective decisions can be made for maximizing the utilization of training complex resources.

The TPF model is capable of assessing the impact of changes in certain input characteristics at the course level, and of showing the effects of these changes at the school and complex levels. Some examples of the course modifications which can be entered into the TPF model are:

a. Annual demand

b. Class capacity

c. Annual number of convenings

d. Course length

e. No-show rates

f. Student failure rate

g. Student setback rate

The model is also capable of assessing the effects of modifications to all courses at a training center, such as might be caused by a change in policy or planning. Examples of 'across-the-board' modifications that may be examined using the TPF model are:

a. Changes in annual demand by a fixed percent of all years of the run

b. Changes in demand on an annual basis; e.g., a growth of 10% per year in demand for each year of the run

c. Changes in the overall center no-show rate

d. Changes in convening frequencies for all courses by a fixed percentage

e. Elimination of all setbacks

f. Introduction of a longer teaching day (result is reduction of course lengths for AOB purposes).

This 'across-the-board' capability is important to the user as the model will only make logical (or rational) changes. That is, fractional course convenings are not allowed, lengths are corrected to multiples of days (or half days by option), and lengths are automatically adjusted for weekend days, without the need for user awareness.

The final output effects of the model are measured in terms of:

a. Average-on-board (AOB)

b. Utilization

c. Throughput

d. Backlog (including average wait time).

Intermediate effects available as model output include:

a. Pass/fail rates

b. Setback rates

c. No-show rates.

MODEL DESCRIPTION The TPF model is an aggregate flow model designed to project the effect on course, school, or training center throughput of changes in demand, capacities, or scheduling. An overview of the TPF system is shown in Figure III-1.

The TPF model will accept data from two sources; first, from the unloaded DOTS Master or Scratch data base, and secondly from data created by a program within the TPF system. This dual input capability is important to the user. The DOTS data base interface allows the user to analyze projections based on the current operational plan, including the latest revisions. The capability of modifying this data base allows the testing of alternate plans within the overall system. The alternate input allows the studying of several alternate plans, without the need to update the data base.

Any input parameter to the model can be considered a variable for manipulation to achieve the desired end results. Any course length, capacity, schedule, demand, or attrition rate can be modified for detailed analysis. The model also allows the operation of a course with one set of characteristics for part of

TPF MODEL – SIMPLIFIED SYSTEM FLOW

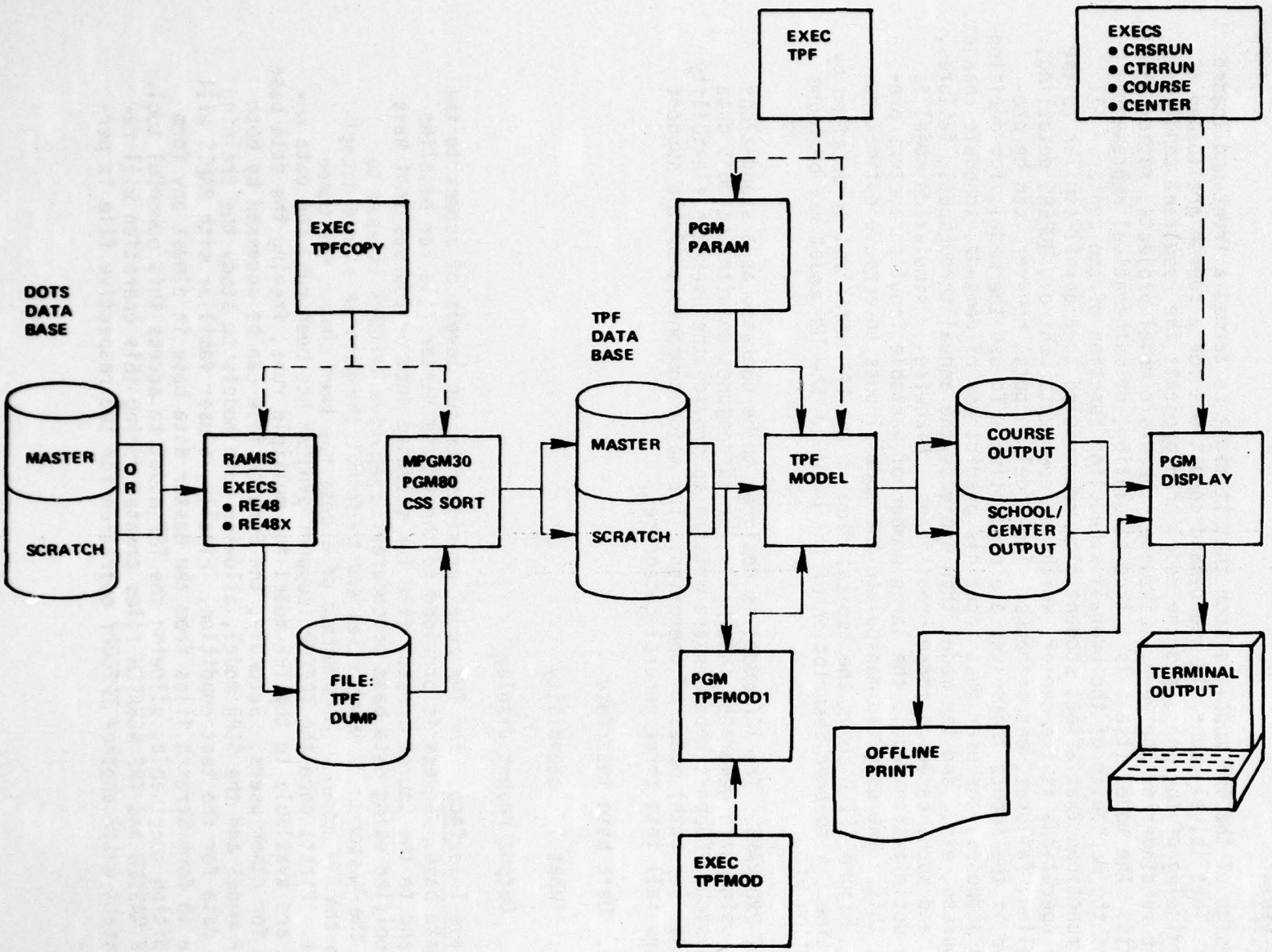


FIGURE III-1

a model run, and an entirely different set of characteristics for the remainder of the run.

The design of the TPF model is such that it might be termed a front-end loaded model. That is, all schedule and demand conditions for a course are loaded at the beginning of the year, and the model then projects the resultant conditions for the remainder of the year. Likewise, the basic model projects, rather than predicts. The model itself is a true simulation, or mathematical representation of the mechanics of the overall scheduling function of the center. All major functions that affect student flow are modeled to a detail similar to the actual operations at the center, which include quota control, BUPERS detailing, scheduling, failures, and setbacks. The model outputs, however, can be predictive if the inputs themselves are predictive. To use the model, for decision-making purposes, the user must input his predictions of demands, schedule changes, new courses, etc., and the model then projects the annual throughputs, failures, AOB's, and such, based on these predictions. Likewise, a simulation model is not an optimization model; the large number of variable input parameters precludes this. The user can manipulate the model inputs to achieve optimal results. Figure III-2 shows the logical flow of the TPF model and is detailed by subroutine. A further description of the logic of the TPF model may be found in SECTION VI.

SUPPORT PROGRAMS. The TPF model is designed to be operated as a stand-alone model, using data generated internally, or in conjunction with the DOTS data base system. Support programs are written to provide the user the flexibility of either (or both) modes of operation. The major components of the support programs fall into three general categories.

a. Data base interface

b. 'What If' capability

c. Output report display

Data Base Interface. The TPF model provides for two levels of access to the DOTS data base. Access is provided to the Course Master file for baseline data, and to the Course Scratch file for selected data or courses that have been modified using data base procedures. Executive TPFCOPY is used to access the Master or Scratch files and to format the data in a form acceptable to the TPF model. This method of access has been chosen for three reasons. First, once the TPFCOPY support program has been run, the data unloaded are available to the TPF model for multiple runs, freeing the data base system for other users. Secondly, the Scratch file can be accessed by both the TPF model and the SCRR model, allowing both models to study the training center data for the test condition. Finally, a user familiar with RAMIS will be able to form Scratch files from the Master data base in almost any form or condition imaginable, allowing the TPF model to access this powerful tool. The TPF MASTER and TPF SCRATCH files created during this operation will remain intact until another TPFCOPY operation for the respective file is performed.

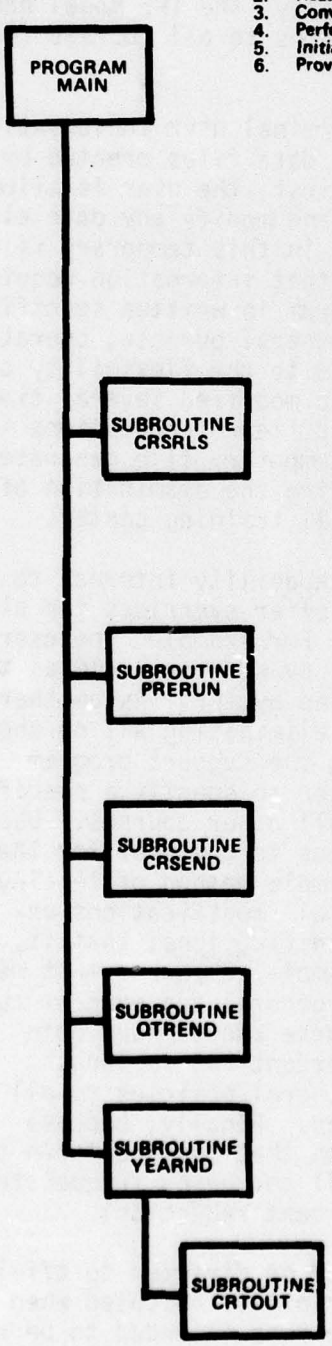
AD-A040 238 IBM CORP CAPE CANAVERAL FLA F/G 5/9
DESIGN OF TRAINING SYSTEMS, TRAPAC USER'S MANUAL, DATA BASE, SC--ETC(U)
SEP 76 K V BRANCH, J D STALEY, R E YANKO N61339-73-C-0097
TAEG-36 NL

UNCLASSIFIED

2 OF 6
AD A040238



TPF MODEL FUNCTIONAL FLOW



1. Read parameter card to determine operations.
2. Read Data-Edit for invalid or missing data (MSG to Printer).
3. Convert course lengths to weeks.
4. Perform "Across the Board" mods and rationalize.
5. Initialize student backlog.
6. Provide following control loops:
   - A. Year
   - B. Quarter (including init loop)
   - C. Week - including:
     - o Data to be used (FY, FY+1, or FY+2)
     - o Rationalize change week and offset
     - o Course release (according to matrix)
     - o Course end calculation

1. Called by MAIN if course is to be started.
2. Calls PRERUN if in 'init' mode to see if course should start.
3. If course is to start:
   - A. Releases BUPERS students, (level-loaded) to class, regardless of over-booking.
   - B. Adds local students (level-loaded) to backlog pool.
   - C. Releases local students from backlog pool to class (up to local capacity).
   - D. Calculates: NOSHOWS = LOCAL STUDENTS* NO SHOW RATE (fractional value carried to next convening).
   - E. Substitute quotas to 10% of class capacity released if space available and course backlogged.
   - F. Multiple convenings of same course up to 10 per week allowed.
   - G. Totals calculated: utilization, convenings, capacity, no shows.

1. Inits model by setting up classes in session at model start.
2. Only starts those courses meeting above criteria.
3. No pools, backlogs, or totals affected.

1. Called by MAIN if course to end in week being calculated.
2. Calculates failures as: FAIL = UTIL * PCT FAIL
3. Adjusts AOB for early disenrollment of failing students.
4. Fractional failures carried forward to next convening.
5. Calculates setbacks as a percentage of failures if there are other sessions of course are running and course length is more than 3 weeks. Adjusts AOB for setback.
6. Totals calculated: Pass, fail, setback, AOB, student days.

1. Adjusts AOB and student days for courses that span quarters.
2. Saves quarterly totals for 'pool' data (backlog, failures, no-shows).
3. Carries forward fractional students from pool to next quarter.

1. Calculates annual totals.
2. Initializes & zeros carry-over elements for multi-year runs.
3. Calculates course, school, & center totals for reports keyed.
4. Calls appropriate report generator routines.

- o Provides center, school, & course reports in a 79 column format for display terminal output or offline printing.

FIGURE III-2

'What If' Support. The 'what if' capability of the model is what allows the TPF system to be a useful tool. The 'what if' capability is supported in three places within the DOTS/TPF system. First, RAMIS can be used to create, modify, or select data to form the Scratch data base that can be used as an input to the TPF model (see Section II, this volume). Secondly, program TPFMOD allows the user to create courses or modify data for existing courses to allow rapid examination of a course under different conditions. Finally, the TPF model has the built-in capability for studying certain modifications to all courses for a run.

TPFMOD is a specialized program intended to allow a terminal user conversational access to data stored in the TPF MASTER or TPF SCRATCH data files created by the COPY operation. The program has two functions. First, the user is allowed to copy all data for any course into a temporary file and modify any data element(s) desired. Secondly, the user can create any new course in this temporary file, using prompting routines specifically requesting only that information required to run the course using the TPF model. Since the program is written specifically to support TPF 'what if' capability and is not general purpose, operation is simple, straightforward, and fast. Two features add to the flexibility of this program. First, the same course may be copied and modified several times, allowing examination of the same course under several different conditions in the same run. Secondly, courses may be created, and the temporary file generated may be concatenated with the Master or Scratch file, allowing the examination of the effects of the addition of the course on the overall training center.

Program PARAM interfaces the user with the 'what if' capability internal to the TPF model. The model allows the user to provide multiplier overrides for all parameters used for calculation within the TPF model. For example, the user may implement a 10% increase in demand for all courses by entering 1.10 as the demand parameter using the support program PARAM, called by TPF. As another example, a training manager could study the effect of eliminating all no-shows by entering 0.00 as the no-show parameter, again using the support program PARAM, called by TPF. This program also allows the user to specify a specific school or course to be executed for a run, bypassing all other courses. Use of this program for 'what if' questions may be advantageous to the user for the following reasons. First, this program is the only simple method of testing the effects of decisions to all courses as it applies all modifications universally. Secondly, this program rationalizes all modifications; that is, it allows only logical modifications to be made. For example, a course that meets once annually will still meet once per year despite a general ten percent cut in convening frequencies, while an eight week long course should have both teaching days and weekend days reduced for a twenty percent cut in length. This program, then, allows the training manager to apply general policies to all courses, maintaining logical conditions for all courses. Finally, because of this the user can test the value of a general action that must be taken to achieve a desired result, (e.g., what general value all convening frequencies must be cut to achieve a center-wide goal of a ten percent reduction).

Display Support. Output generated by the TPF model may be directed to off-line printers or terminal displays. The use of off-line printers indicated when large amounts of data are to be displayed or base line runs intended to be retained are generated. A display support program was developed to allow the

user to browse through the data files created by the TPF output, displaying only those data required. Course data created can be called by CDP number, while all output, including summary reports, may be called by page number. A LOOK option allows the user to look at the first line of each page in the file to ascertain if display of the entire page is desirable.

INPUT PARAMETERS DESCRIPTION

All baseline data for the TPF model are stored in the DOTS Master data base course file. Section II of this volume describes the procedures for loading and updating these data. A listing of these data elements and their format is shown in Table III-1. The data used as input to the TPF model are compatible with NITRAS data elements except for those elements where data unique to the center under study are required. The data elements are:

a. CDP - The CDP code is the unique 4 position alphanumeric identifier code for each course taught in the Navy (from NITRAS). This code is used as the sort key for data base operations.

b. SCH - This is a four digit code to describe the school within the training complex to which the course is assigned. Currently 4 descriptive characters such as AC&R for the Air Conditioning and Refrigeration school are used for this code.

c. DEL - The delete flag is set automatically by the data base maintenance routines and is used to signify courses in the file that have either been disestablished or are not yet taught at the Center.

d. CTLG - This is the 10 position alphanumeric CIN code used in the Catalog of Navy Training Courses (CANTRAC). This code is identical for the same course taught at different locations (from NITRAS).

e. CRS - This is the 16 character name of the course (Activity Short Name from NITRAS).

f. CONV1 - This is the annual number of convenings for the course planned for the current fiscal year (from NITRAS).

g. LEN1 - This is the length in days including weekend days, of the course for the current fiscal year (from NITRAS).

h. CAP1 - This is the current limiting capacity for the course (from NITRAS).

i. BCAP1 - This is the current number of seats in the course allocated to BUPERS or other agencies and not available for local quota control (locally obtained).

j. DMD1 - This is the current total anticipated annual demand for the course (from NITRAS).

k. BDMD1 - This is the current annual demand for the course for students detailed by BUPERS and other agencies (locally obtained).

RAMIS DUMP FORMAT (MASTER OR SCRATCH)

± EBCDIC RECORD LENGTH 80 CHARACTERS

FIELD NAME	FORMAT		LENGTH
CDP	A	4	4
SCH	A	4	4
DEL	A	1	1
CTLG	A	10	10
CRS	A	16	16
CONV1	I	3	3
LEN1	F4	.1	4
CAP1	I	3	3
BCAP1	I	3	3
DMD1	I	5	5
BDMD1	I	5	5
OFST	I	2	2
PFAIL	F5	.3	5
NADIS	F5	.3	5
NOSHOW	F5	.3	5
SETB	F5	.3	5
		X	0
BKLG	I	4	4
JNO	A	4	4
CONV2	I	3	3
LEN2	F4	.1	4
CAP2	I	3	3
BCAP2	I	3	3
DMD2	I	5	5
BDMD2	I	5	5
CHWK2	I	3	3
CONV3	I	3	3
LEN3	F4	.1	4
CAP3	I	3	3
BCAP3	I	3	3
DMD3	I	5	5
BDMD3	I	5	5
CHWK3	I	3	3
PAOB	I	4	4
AFAIL	F5	.3	5
PRTY	I	2	2
CTYPE	A	2	2
PNEC	A	4	4
UCAP	I	3	3

TABLE III-1

l. OFST - Offset is a model unique value and is used to modify the schedule matrix to allow the model to convene courses with low convening frequencies (3 or less) in the proper quarters. The scheduler matrix normally schedules the first convening of a course for the first or second week of the model run. An offset of 26 would change the first convening to the 27th week of the model (3rd quarter). Use of offset is optional and does not affect annual totals.

m. PFAIL - This is the current historical (last 12 months) failure rate for the course (from NITRAS).

n. NADIS - This is the historical (last 12 months) non-academic disenrollment rate for the course (from NITRAS).

o. NOSHOW - This is the historical (last 12 months) no-show rate experienced by the course. This rate is determined as the percentage of those scheduled for the course that did not attend or cancel (locally obtained).

p. SETB - This is the historical (last 12 months) set back rate for the course. This value is not used for model calculations (from NITRAS).

q. BKLG - This is the length in weeks that a student must wait for a local quota for a course. Optionally, the model may be configured to accept backlog in students rather than weeks. However, usage must be consistent (locally obtained).

r. JNO - This is the obsolete 4 position J number used as an identifier for courses. It is retained as a cross-reference to statistical data on file. New or modified courses do not have a J number (not updated).

s. CONV2 - Same as CONV1 but for first out year (*).

t. LEN2 - Same as LEN1 but for first out year (*).

u. CAP2 - Same as CAP1 but for first out year (*).

v. BCAP2 - Same as BCAP1 but for first out year (*).

w. DMD2 - Same as DMD1 but for first out year (*).

x. BDMD2 - Same as BDMD1 but for first out year (*).

y. CHWK2 - This term is used to force the week in which the first out year values will be used for a course. During normal processing the first out year values will be activated in the 53rd week. A non-zero entry in CHWK2 will override this, and the value entered will be used.

(*) NOTE - For items marked (*) the model will default and use current (initial) values if the data for the field is zero or missing. However, use of CHWK2 will override this and the user must provide all first out year data.

z. CONV3 - Same as CONV1 but for second out year (**).

aa. LEN2 - Same as LEN1 but for second out year (**).

bb. CAP3 - Same as CAP1 but for second out year (**).

cc. BCAP3 - Same as BCAP1 but for second out year (**).

dd. DMD3 - Same as DMD1 but for second out year (**).

ee. CHWK3 - Same as CHWK2 but for second out year data.

(**) NOTE - For items marked (**) the model will default and use current (or first out year values if CHWK2 is used) if the data for the field is zero or missing. However, use of CHWK3 will override this and the user must provide all second out year data.

ff. PAOB - This is the planned AOB for the course and is not used in model computations (from NITRAS).

gg. AFAIL - This is the planned (or allowed) failure rate for the course and is not used in model computations (from NITRAS).

hh. PRTY - This is the 2 digit course priority code (from NITRAS).

ii. CTYPE - This is the 2 position type code for the course (i.e., fleet, A, C, etc.) (from NITRAS).

jj. PNEC - This is the primary NEC code for those courses granting NEC's (from NITRAS).

OUTPUT PARAMETERS DESCRIPTION

The primary TPF model output is the yearly course summary format, shown in Figure III-3a. This format is broken into two parts; the upper portion which gives a synopsis of the input data, and the lower position which tabulates the fiscal year projections based on the input data. The data elements on the top have been previously described. The following discussion will, therefore, concentrate on the yearly course projections.

a. FORMAT - The output is broken into 4 quarters of equal length (13 weeks). The first quarter is assumed to start 1 July of the fiscal year. This is significant as the model will not run courses during the Christmas period, which is assumed to be the last week of the second quarter and the first week of the third quarter. Annual totals as well as percentages, are printed for applicable data elements.

b. NUMBER OF CONVENINGS - A convening date is defined as the day of the first class for a course section. Thus, a convening begins accumulating within a quarter on the day it begins. For example, if a course with a length of 11.0 weeks convened in the 13th week of the 1st quarter, the convening will be counted in the first quarter, even though the bulk of the student AOB falls in the second quarter. A

0284 FINANCIAL MGMT

SCHL DATE CATALOG NO NEC
S/DA 0812 A-551-0050

P AOB PFAIL TYPE PRTY
2.0 7.0% F1 31

--SCHEDULE DATA--	FY	FY+1	FY+2
ANNUAL CONVENINGS	13	0	0
COURSE LENGTH DYS	5.0	0.0	0.0
COURSE CAPACITY	25	0	0
BUPERS CAPACITY	0	0	0
ANNUAL DEMAND	150	180	180
BUPERS DEMAND	0	0	0

CRS BACKLOG
0 STU 32
0 WKS 5

FY+1 WK 0
FY+2 WK 0
OFFSET 0

QUARTERLY PROJECTIONS

	1 QR FY 75		2 QR FY 75		3 QR FY 75		4 QR FY 75		ANNUAL TOTAL	
0 CONVENINGS	4		3		3		3		13	
CRS CAPACITY	100		75		75		75		325	
UTILIZATION	67	67.0%	30	40.0%	29	38.7%	30	40.0%	156	48.0%
NO SHOWS	11	14.1%	5	14.3%	5	14.7%	5	14.3%	26	14.3%
BACKLOG 0/WKS	0	0 WKS	0	0 WKS	0	0 WKS	0	0 WKS	0	0 WKS
STU DAYS/AOB	335	3.7	150	1.6	145	1.6	150	1.6	780	2.1
TOTAL PASS	62	92.5%	27	90.0%	25	86.2%	26	86.7%	140	89.7%
TOTAL FAIL	5	7.5%	3	10.0%	3	10.3%	3	10.0%	14	9.0%
SETBACK ACAD	0	0.0%	0	0.0%	0	0.0%	0	0.0%	0	0.0%
DISENROLL NA	0	0.0%	0	0.0%	1	3.4%	1	3.3%	2	1.3%

TPF MODEL -- FISCAL YEAR 1976 SUMMARY
SCHOOL CODE COMM

COMM SCHOOL SUMMARY (RUN 081275)

	1 QR FY 76		2 QR FY 76		3 QR FY 76		4 QR FY 76		ANNUAL TOTAL	
0 CONVENINGS	47		41		44		41		173	
CRS CAPACITY	683		589		645		589		2506	
UTILIZATION	602	88.1%	447	75.9%	441	68.4%	383	65.0%	1873	74.7%
NO SHOWS	77	11.3%	68	13.2%	64	12.7%	58	13.2%	267	12.5%
BACKLOG 0/WKS	181	7 WKS	90	6 WKS	46	5 WKS	27	3 WKS	27	3 WKS
STU DAYS/AOB	2810	3.8	2052	22.5	2051	22.5	1661	18.2	8574	23.5
TOTAL PASS	588	97.7%	434	97.1%	430	97.5%	372	97.1%	1824	97.4%
TOTAL FAIL	14	2.3%	13	2.9%	11	2.5%	11	2.9%	49	2.6%
SETBACK ACAD	0	0.0%	0	0.0%	0	0.0%	0	0.0%	0	0.0%
DISENROLL NA	0	0.0%	0	0.0%	0	0.0%	0	0.0%	0	0.0%

TPF MODEL -- FISCAL YEAR 1976 SUMMARY
** TOTALS FOR TRAPAC TRAINING CENTER **

CENTER CODE N SUMMARY (RUN 081275)

	1 QR FY 76		2 QR FY 76		3 QR FY 76		4 QR FY 76		ANNUAL TOTAL	
0 CONVENINGS	759		695		678		720		2852	
CRS CAPACITY	23900		22514		21648		24265		92335	
UTILIZATION	16209	67.8%	12559	55.8%	11825	54.6%	12918	53.2%	53511	58.0%
NO SHOWS	2423	13.0%	1974	13.6%	1861	13.6%	2082	13.9%	8340	13.5%
BACKLOG 0/WKS	3432	9 WKS	2676	11 WKS	2325	13 WKS	2172	15 WKS	2172	15 WKS
STU DAYS/AOB	92066	10.9	73855	89.4	71520	783.8	73931	81.2	311372	853.1
TOTAL PASS	15686	96.8%	12869	96.1%	11348	96.0%	12393	95.9%	51496	96.2%
TOTAL FAIL	417	2.6%	375	3.0%	365	3.1%	400	3.1%	1557	2.9%
SETBACK ACAD	36	0.2%	27	0.2%	27	0.2%	30	0.2%	120	0.2%
DISENROLL NA	106	0.7%	115	0.9%	112	0.9%	125	1.0%	458	0.9%

FIGURE III-3 SAMPLE COURSE, SCHOOL, AND CENTER REPORTS

course convening is based on the current week, offset, and change week.

c. COURSE CAPACITY - This represents a total of the limiting course student capacity on the day each course was released, by quarter. On the day a course convenes, the model tests if this convening is prior to or after a schedule change date, and the current capacity is used.

d. UTILIZATION - This is the total number of class seats occupied on the day the course convened. That is, it is a total of the local quotas granted, minus no-shows, plus BUPERS input, plus substitute quotas. This algorithm does make allowance for overbooking of critical courses.

e. NO-SHOWS - This is an historical average of students not showing up for the course who had previously been granted quotas. Only that portion of students scheduled through local quota control is counted in the no-show figure. Students scheduled from BUPERS do not contribute to no-shows in this model as: (1) they are usually under TAD orders; and (2) the school has no record of these students until they arrive. No-shows also are not derived from substitute quotas for obvious reasons. Based on this use of no-show rates, it is important that the historical no-show rate entered in the model be based on the sum of no-shows and utilization, and not on utilization alone.

f. BACKLOG - Backlog is defined as the time a student must wait for a confirmed quota in a course that is booked. The backlog printout is a snapshot of the actual backlog on the last day of the quarter. The model input portion of the format gives the initial, or beginning backlog, while that contained in the annual totals column is the ending backlog, which is identical to the 4th quarter backlog. Backlog is entered in weeks and is converted to students who hold quotas. The output report shows backlog in both weeks and numbers of students.

g. STUDENT DAYS - Student days are the total, by quarter, of the number of days each student has spent in a course that quarter. Thus, it is possible to show student days in a quarter that had no convenings. Students set back to a course do not count as utilizations, but the student days spent in that convening are totaled. Two methods of calculating student days are available in the TPF model, with the method decided by the value of AOB constant entered. If this constant is less than 300, student days is the total of class days times students, adjusted for setbacks and quarters. If the AOB constant is 300 or greater, student days include weekend days for courses longer than one week. If a course spans two quarters, the weekend days for that week the span occurred are charged to the quarter in which the course convened. AOB is the student days divided by the AOB constant, which is a model input parameter. This AOB is multiplied by 4 for the quarterly outputs. The student days output will be printed as K student days (student days/1000) if the parameter exceeds 99,999 student days. The format will read KSTU DAY rather than STU DAYS to indicate this.

h. PASS/FAIL, ETC. - These are historical pass/fail and non-academic disenrollment percentage rates. Updates to these factors require additional statistical analysis of current data. Academic setbacks are hard coded into the logic of the model and are based on the fail percentages. The criteria for setbacks are discussed under the logic design portion of this section.

School and center summary reports are also available as outputs of the TPF model. The data presented are a summary, by school code or by center, of the data presented in the individual course reports. Input data are not displayed for summary reports. The backlog wait time in weeks presented in the summary reports is the average wait time only for those courses having a backlog. Samples of the course, school, and center formats are presented in Figure III-3b and III-3c.

TPF MODEL SYSTEM OPERATIONAL PROCEDURES

GENERAL SYSTEM FLOW. The Training Process Flow system can run using data either from the unloaded DOTS data base, or any combination of courses from the unloaded data base, modifications of these courses, or new courses entered by the user. It is assumed that the user is familiar with the data base operating procedures, including modification of these data to produce a Scratch data base.

The basic flow of the TPF model system was shown in Figure III-1. The model can be run under baseline conditions (with the output usually saved) or with the input data modified in order to study courses under various conditions. These input data can be modified at three points in the TPF system. First, all course data elements can be modified using facilities of the data base update/maintenance routines. The user is directed to Section II, DATA BASE MAINTENANCE, for a full description of the facilities available. Use of these facilities is recommended when a large number of changes are to be made, especially if the changes are to be made only if certain select criteria are met; e.g., modify the convening frequency of a course only if the length is greater than two weeks. Secondly, data can be modified using the facilities of the TPF support program TPFMOD which allow selection, revision, and creation of courses for direct input to the TPF model. Use of program TPFMOD is indicated when the number of changes is small, the same course is to be studied under several different configurations, or new courses are to be studied. Finally, capability exists within the TPF model to run only selected courses or schools and to make across-the-board modifications, such as reducing the no-show rate for all courses by 10 percent. These modification features can be used individually, or in any combination, to allow the user a flexible, easy to use model system.

Data Base Interface. All baseline data used as input to the TPF model are stored in the DOTS data base. These data are updated periodically so as to be current with actual conditions and should be considered the master source file. However, the TPF model cannot access these data directly. In order to be accessible to the TPF model, the course file must be unloaded using procedures contained in this section. Prior to this process, the unloaded data may be modified, forming a Scratch data base, using data base operating procedures. It is necessary to copy these data to the TPF system. If unmodified data are to be used as the TPF Master data base, enter:

TPFCOPY MASTER

If the message:

```
15.26.43    TPFCOPY MASTER

** FOR INFORMATION ON THE SEPTEMBER RAMIS USERS GROUP MEETING,
**  F O R U M ,  PLEASE TYPE    HELP FORUM .

** NEW RAMIS DOCUMENTATION IS READY. PLEASE TYPE HELP DOCUMENT
** FOR ADDITIONAL INFORMATION.

TPF MASTER DUMPED BY RAMIS
EXECUTION:

     594 : TOTAL INPUT DUMP RECORDS
     297 : TOTAL OUTPUT COMBINATION RECORDS
TPF      DUMP      P      594   F 80    08/31/76   08/31/76
TPF      SORT      U      297  F 160    08/31/76   08/31/76
TPF      WORK      P      297  F 160    08/31/76   08/31/76
     297 : INPUT RECORDS PROCESSED
     594 : OUTPUT RECORDS PROCESSED
TPF    MASTER    P      594   F 80    08/31/76   08/31/76
```

is displayed, the data base was successfully copied and will be available as the baseline or master input to the TPF model system. If an error occurs during unload, a message will indicate whether RAMIS encountered an error during unload or the data base is not on line.

If it is desired to copy a Scratch data base for model manipulation, enter:

TPFCOPY SCRATCH

The error messages are the same, and the word SCRATCH will be substituted for MASTER if the unload is successful.

During the data base unload operation, core storage must be set to 320K. If this is not done, the system will automatically set core to 320K. However, this causes the executive to terminate and the user must restart the copy operation.

TPF Course Create/Modify Program. The TPF model can execute using the TPF Master or Scratch data base directly, or program TPFMOD can be invoked to further modify the data in either the Master or Scratch data base, including the addition of new courses. This program should be used only if modifications to individual courses are required or new courses are to be entered. Only those courses copied or created will appear as TPF input if this program is invoked. If the Master TPF data base is to be used as reference, enter TPFMOD MASTER to invoke the program. If the Scratch data base is to be used, enter TPFMOD SCRATCH. The message:

TPF CREATE/MODIFY- ENTER NEW OR COPY

will be displayed.

Options. Three basic OPTIONS exist in the TPFMOD program; NEW, COPY, and MOD.

NEW - The NEW option signifies that a new course is to be created. The program prompts for the entries necessary to create a new course. Upon completion of the prompted data entry, the program allows the user to add further data or to modify data already entered. Once DONE is entered, the creation of the course terminates and the system requests the next user option.

COPY - The COPY option allows the user to copy any course from the Master (or Scratch) data base for entry to the TPF model. The CDP number of the course must be entered and the message:

CDP AAAA COPIED- ENTER DONE OR MOD

will appear if the course was found in the file and copied. The message:

AAAA +NOT IN FILE - RETRY OR ENTER DONE

appears if the requested CDP was not in the file.

Once copied, the user can modify any course data element or proceed to any other TPFMOD option. It is possible, using the COPY option, to have multiple copies of the same course in the run file, each modified to study different configurations during a model execution.

MOD - The MOD option allows the user to modify data stored in the run file during this execution. The MOD option requests:

ENTER CDP

If a carriage return is entered, the last course copied or created will be modified. If a CDP code is entered, the CDP will be modified. If multiple copies of a CDP are in the file, only the first entered will be available for modification. If the CDP number entered is not in the file, the message:

XXX INVALID- RETRY

will be displayed.

If the CDP is located, the message

MODIFY DATA FOR CDP XXXX

will be displayed.

At that time, any of the 36 TPF fields may be modified. A list of the fields and their formats appears in this section, and the list may be displayed on the terminal by entering HELP. To modify a field, the user should enter the field name. The system then responds:

ENTER FIELDNAME (TYPE LEN)

The data entered will be 'echoed' if accepted, or an appropriate error message will be displayed. Any number of fields may be modified in this mode. Enter DONE or QUIT to exit this mode.

Fields and Formats. The 36 fields that may be modified using program TPFMOD are identical to those noted in the section describing the TPF model. The command HELP may be entered any time during execution for display of the fields. The fields available for modification are:

LIST	FIELD NAME	ALIAS	TYPE	LENGTH
1	CDP	CDP	A	4
2	CTLG	CTLG	A	1Ø
3	CRS	CRS	A	16
4	CONV	CONV1	I	3
5	DAYS	LEN1	F	4.1
6	CAP	CAP1	I	3
7	BCAP	BCAP1	I	3
8	DMD1	DMD1	I	5
9	BDMD1	BDMD1	I	5
1Ø	OFST	OFST	I	2
11	PFAIL	PFAIL	F	5.3
12	NADIS	NADIS	F	5.3
13	NOSHOW	NOSHOW	F	5.3
14	SETB	SETB	F	5.3
15	BKLG	BKLG	I	4
16	JNO	JNO	A	4
17	CONV2	CONV2	I	3
18	LEN2	LEN2	F	4.1
19	CAP2	CAP2	I	3
2Ø	BCAP2	BCAP2	I	3
21	DMD2	DMD2	I	5
22	BDMD2	BDMD2	I	5
23	CHWK2	CHWK2	I	3
24	CONV3	CONV3	I	3
25	LEN3	LEN3	F	4.1
26	CAP3	CAP3	I	3
27	BCAP3	BCAP3	I	3
28	DMD3	DMD3	I	5
29	BDMD3	BDMD3	I	5
3Ø	CHWK3	CHWK3	I	3
31	PAOB	PAOB	I	4
32	AFAIL	AFAIL	F	5.3
33	PRTY	PRTY	I	2
34	CTYPE	CTYPE	A	2
35	PNEC	PNEC	A	4
36	SCH	SCHOOL	A	4

The program prompts for data during new course generation or field modification in the form:

ENTER FIELDNAME (TYPE LEN)

where FIELDNAME is the abbreviated name for the field, TYPE contains an A for alphanumeric data, I for numbers with no decimal, or F for data with a decimal point, and LEN is the length of the data entry (number of characters). Appropriate error messages will be displayed if the entry is of the wrong type or invalid for the field in question.

TPFMOD Commands. The following commands are available to the user of the TPFMOD program:

QUIT	DONE	HELP
STAT	SHOW	COPY
NEW	MOD	

The function of the COPY, NEW, and MOD commands has already been described. The functions of the other commands are as follows:

QUIT - The QUIT command terminates execution of the TPFMOD program. The user is then asked whether the run data set created is to be saved. An N reply will terminate the program without saving the created data set, while a Y reply will over write the previous TPF run data set. At this point, any of the other commands are available to reenter the program.

DONE - The DONE command terminates the current step in the program, and causes a return to the control portion of the program.

HELP - The HELP command allows the user to display a list of the valid TPF fields or valid system commands. The system returns to the point at which the HELP command was entered upon completion of this display.

STAT AAAA - The STAT command will display the status of field AAAA of the current course under consideration (course being modified or last course copied or modified). AAAA represents the field name abbreviation of the field to be displayed.

SHOW - The SHOW command displays all fields for the current course under consideration. An option exists with SHOW to display a single field by entering the field name when requested. A carriage return (with no entry) will cause all fields to be displayed. The system returns to the point at which the SHOW command was entered upon completion of this display.

TPF Model. All TPF model execution is under the control of the user and may be modified by the user at run time. Parameters that can be modified at this time include input data selection, output report generation, and modification of selected data elements.

The program is called by entering:

TPF xxxx yyyy

where:

xxxx is the data base to be used as the input data set. If this field is MASTER, the Master data base will be used for data input; if the field entry is SCRATCH, the Scratch data base, created by program TPFCOPY, will be used. This field should be TEST if output created by TPFMOD is to be used as input.

yyyy field is optional and it may contain the word SAVE. SAVE specifies that the output of this run should be stored in a save area. A blank or invalid entry will cause output to be directed to normal output areas. This option is normally used to preserve the results of baseline runs.

Control Fields. The following field options are available during this phase to control TPF execution. Default values are used for the first four fields if the entry is invalid or missing. The control fields and their use are:

INPUT - This field determines whether the input is to be from the TPF Master/Scratch data base or from the data created by the TPFMOD program.

A 10 in this field will cause the model to use data from the Master, Scratch, or TPFMOD data base, while a 5 indicates that the input is to be from card or tape. (Default 10)

FYR - Indicates the year and data to be used as the first year of the model run; e.g., 1976. (Default 1976)

LYR - Indicates the last year of the model run. The difference between these years, plus 1, indicates the number of years to be projected, with a maximum of three. (Default 1976)

AOB - This indicates the number of days to be used in the AOB calculation and controls the method. The standard number to be used is 365 causing weekend days to be used in the calculation. If a figure less than 300 is used, weekend days will not be counted in AOB calculations. (Default 0365)

PRINT - A 0 or blank in this field will cause all output to be directed to data sets for terminal or CRT display, or optional offline printing.

INIT - A 1 in this field will cause the model to initialize those courses in session at model start, while a 0 or blank will skip this initialization. The model should be initialized for all normal runs.

Data Fields. The following fields are available to modify input parameters for all courses during the model run, with no changes to any data base used. All inputs should be four characters in length including decimal points e.g., 1.25 or .931. Minus signs or alpha characters are not permitted.

FAIL - All failure rates are multiplied by the number entered to produce the new failure rate.

NOSHO - All no-show rates are multiplied by the number entered to produce new no-show rates.

DMD - All demands, both bureau and local, for all three years, are multiplied by the number entered.

TREND - All local and bureau demands for the second and subsequent years are multiplied by the number entered to represent trends; e.g., 10 percent growth per year.

CONV - All convening frequencies are multiplied by the number entered. Only whole number changes are allowed; fractional convenings are counted as the next higher number. A 0 in this field is ignored.

LEN - All lengths are multiplied by the number entered. Only half-day increments are allowed, and compensation is made for weekend days.

CAP - All capacities are multiplied by the number entered. Only whole number changes are allowed; fractional capacities are counted as the next higher number.

SETB - All setback rates are determined by using this number as the percentage of students failing that are candidates for setback. The model calculates actual setbacks only if courses are greater in length than two weeks and multiple convenings occur.

Report fields. A 1 in fields CRS, SCHL, or CTR will cause the respective course, school, or center reports to be produced, while a 0 will cause bypassing of these reports. All zeros in these fields will cause the model to cycle with no reports produced.

Select Fields. The following two fields can be used to select courses as input to the TPF model.

CDP - Only a course with the entered CDP number will be executed, with all others bypassed.

SCH - Only courses matching the school code entered will be executed, with all others bypassed. This select requires an absolute match between the data in the select field and that in the data base. However, if the select field uses $'s in any position, testing of that position is bypassed. For example, an entry of 9$$$ would produce a report containing all courses with a first character of 9. The $ option may be used in any or all fields of this entry.

TPF Operational Flow. Following entry of the command TPF xxxx, the message:

TPF MODEL - WANT INSTRUCTIONS (Y OR N)

will be displayed. If a Y is entered, the list of instructions and fields accessible will be displayed; if an N is entered, this step is by-passed. Next, the present run conditions are displayed, followed by the request:

ENTER RUN, QUIT, OR FIELD TO BE CHANGED

If RUN is entered, the model will execute under the present condition, while if QUIT is entered, execution will be by-passed. To activate any of the previously mentioned fields, enter:

	FIELDNAME	DATA
e.g.,	FAIL	.941

To deactivate a previously selected field, enter:

	FIELDNAME	DEACT
e.g.,	FAIL	DEACT

TPF Commands. The following commands are available during the execution of this program:

RUN - Causes the model to execute using the currently selected conditions.

QUIT - Causes termination of the program and by-passes model execution.

SHOW - Causes the current run condition to be displayed.

HELP - Causes the program instructions, fields, and formats to be displayed.

Output Display. Following a successful execution of the TPF model, the message:

END TPF EXECUTION
ENTER CTRRUN FOR CENTER FORMAT: CRSRUN FOR COURSE FORMAT

will be displayed.

The program that displays these outputs uses page control, and displays a message at the beginning of the program as to the type of file and number of pages in the file.

If a school/center summary is to be displayed, enter CTRRUN (CENTER for SAVE option). When the display program requests the page number, enter either the page number to select the desired page or a 0, or carriage return to display the next page. The terminal will pause at the end of each page prior to the next entry.

If a course file is to be displayed, the user has the option to select pages by CDP or page number. To display the course file, enter CRSRUN (COURSE for SAVE option). The message:

COURSE FILE - SELECT BY CDP OR PAGE NO.

will be displayed. If a page number is entered, that page will be displayed, while a carriage return will display the next page. If CDP is entered, the system will ask for the CDP number to be displayed. If the CDP number entered is not in the file, an error message will be displayed.

To terminate the display program, enter DONE or QUIT.

Display Commands. The following commands are available during the display option:

QUIT - Terminate the display and exit.

DONE - Same as QUIT.

CDP - If a course file, allows page selection by CDP.

PAGE - Allows selection by page number. Optionally, the page number can be included in the page request.

LOOK NNN - Allows the user to look at the first line of page NNN.

TPF Output Formats. Figures III-3a, b, and c show samples of the course, school, and center formats produced by the TPF model.

OPERATIONAL EXAMPLE. The following problem will illustrate an application of the TPF model to aid training complex management in assessing the feasibility of meeting various objectives placed by the functional command. In this scenario, training management has been asked to:

a. Hold AOB for the complex at a near 800 level

b. Reduce the no-show rate to a 10% level

c. Release 7% of their instructor billets to the functional command

The first step in evaluating various strategies is to obtain a baseline run to establish current conditions. The use of the TPF model to accomplish this is shown in Figure III-4. The first step is to unload the current data base. To accomplish this core storage is set to the required 320K and then the command TPFCOPY MASTER is entered. (all user inputs are underlined in the following examples). The program executes and the verification message is displayed, indicating 135 courses have been unloaded from the master data base. The next step is to run the TPF model. The user enters TPF MASTER SAVE in order to use the master data base and to save the output for further reference. The Y response to the first program query by the user caused the listing of the fields available for modification; the usual response is N here. The display

COPY MASTER DATA BASE

```
          SET CORE 32Ø
CSS.212 Ø9/23/75

12.23.51 TPFCOPY MASTER

** FOR INFORMATION ON THE SEPTEMBER  RAMIS USERS GROUP MEETING,
**  F O R U M ,  PLEASE TYPE  HELP FORUM .

** NEW RAMIS DOCUMENTATION IS READY. PLEASE TYPE HELP DOCUMENT
** FOR ADDITIONAL INFORMATION.

TPF MASTER DUMPED BY RAMIS
EXECUTION:

     594 : TOTAL INPUT DUMP RECORDS
     297 : TOTAL OUTPUT COMBINATION RECORDS
TPF     DUMP      P      594   F 8Ø    Ø8/31/76   Ø8/31/76
TPF     SORT      U      297   F 16Ø   Ø8/31/76   Ø8/31/76
TPF     WORK      P      297   F 16Ø   Ø8/31/76   Ø8/31/76
     297 : INPUT RECORDS PROCESSED
     594 : OUTPUT RECORDS PROCESSED
TPF     MASTER    P      594   F 8Ø    Ø8/31/76   Ø8/31/76
```

RUN TPF MODEL

```
12.24.36 TPF MASTER SAVE

INPUT THIS RUN:
MASTER
SAVE OPTION IN EFFECT THIS RUN
ENTER CARRIAGE RETURN TO CONTINUE

TPF MODEL - WANT INSTRUCTIONS (Y OR N)
_Y

ENTER: FNAME XXXX  (SET FNAME TO VALUE)
       FNAME DEACT (DEACT FNAME)
       SHOW        (SHOW RUN COND)
       RUN         (RUN MODEL)

FIELDS:  FYR   (NNNN)  LYR   (NNNN)  AOB   (NNNN)  CDP   (AAAA)  SCH   (AAAA)
         FAIL  (NNNN)  NOSHO (NNNN)  DMD   (NNNN)  TREND (NNNN)  CONV  (NNNN)
         LEN   (NNNN)  CAP   (NNNN)  SETB  (NNNN)

REPORTS: CRS   (N)     SCHL  (N)     CTR (N)       PRINT (N)

PRESENT RUN CONDITIONS
REPORTS:  CRS   CTR
INPUT = 1Ø     FYR   = 1976     LYR   = 1976     AOB   = Ø365

ENTER RUN, QUIT, OR FIELD TO BE CHANGED

ENTER REQUEST:
_CRS DEACT

ENTER REQUEST:
_SHOW

PRESENT RUN CONDITIONS
REPORTS:  CTR
INPUT = 1Ø     FYR   = 1976     LYR   = 1976     AOB   = Ø365

ENTER REQUEST:
_RUN
+++END TPF EXECUTION
ENTER: CENTER FOR CENTER SUMMARY; COURSE FOR COURSE OUTPUTS
12.23.55
```

FIGURE III-4 TPF MODEL BASELINE RUN

DISPLAY RESULTS

```
          CENTER
STANDARD FILE - 21 LINES
  4 PAGES IN FILE

+ENTER PAGE NUMBER
1
PAGE   1
                        TPF MODEL -- FISCAL YEAR 1976 SUMMARY
                      ** TOTALS FOR TRAPAC TRAINING CENTER **

CENTER CODE N  SUMMARY  (RUN #81275)

            ! 1 QR FY 76 ! 2 QR FY 76 ! 3 QR FY 76 ! 4 QR FY 76 ! ANNUAL TOTAL
------------|------------|------------|------------|------------|--------------
# CONVENINGS!   759!       !   695!       !   678!       !   72#!       !   2852 !
CRS CAPACITY!23998!       !22514!       !21648!       !24265!       ! 92335 !
UTILIZATION !16249!  67.8%!12559!  55.8%!11825!  54.6%!12918!  53.2%! 53511 !  58.#%
NO SHOWS    ! 2423!  13.#%! 1974!  13.6%! 1861!  13.6%! 2#82!  13.9%!  834#  !  13.5%
BAKLOG #/WKS! 3432! 9 WKS! 2676!11 WKS! 2325!13 WKS! 2172!15 WKS!  2172  !15 WKS
STU DAYS/AOB!92519!1#13.9!74192! 813.1!71859! 787.4!74288! 814.1!312849  ! 857.1
            !      !      !      !      !      !      !      !      !        !
TOTAL PASS  !15686!  96.8%!12#69!  96.1%!11348!  96.#%!12393!  95.9%! 51496  !  96.2%
TOTAL FAIL  !  417!   2.6%!  375!   3.#%!  365!   3.1%!  4##!   3.1%!  1557  !   2.9%
SETBACK ACAD!   36!   #.2%!   27!   #.2%!   27!   #.2%!   3#!   #.2%!   12#  !   #.2%
DISENROLL NA!  1#6!   #.7%!  115!   #.9%!  112!   #.9%!  125!   1.#%!   458  !   #.9%
-------------------------------------------------------------------------------
PAGE   1  +PAUSING- ENTER NEXT PAGE NO.
DONE
+DISPLAY TPF ENDED

12.3#.48
```

FIGURE III-4 (Cont) TPF MODEL BASELINE RUN

also shows that a course report is keyed as well as the center summary report. Only the summary report is required so the course report is deactivated. The next step is to verify run conditions, and if satisfied run the model. Following model execution CENTER is executed to display the results which are shown at the location of Figure III-4.

The current AOB is projected by the TPF base run is about 857. To meet the near 800 objective, two alternatives are available; refuse training or reduce the length of the training. To test the sensitivity of the system to a reduction in length, all courses will be reduced in length by 10%. To simulate the new no-show rate, the current rate must be cut by 20%. The operational steps to accomplish this are shown in Figure III-5. The Master data base need not be recopied as it is not modified by TPF runs. The first step then is to call the TPF model and enter the parameter overrides. Following execution CTRRUN is used to display the results obtained. By observation, it can be seen that the reduction in AOB can be achieved by a 10% reduction in course lengths. (In practice this might include shortening of some courses and extending the teaching day in others). The effects upon throughput and no-show rate can also be observed.

In order to achieve a reduction in instructor billets, a cut in convening frequency is one alternative. In the third part of this example (Figure III-6). a reduction in convening frequencies by 10% was attempted. However, the results of this effort show that the number of students backlogged increased to an unacceptable 8% level, the trend for the last quarter is increasing, and the wait time is increasing. The results of this run showed that a 10% reduction in convenings is not a feasible alternative.

In summary, this example shows an application of the TPF model to a practical problem. The training manager could rapidly test the effectiveness

of different strategies, and would most likely further test alternative strategies. By use of the TPF 'across-the-board' capability, the user did not need to consider the possibility of fractional convenings or adjusted course lengths; the model performed these rationalizations automatically. The model likewise does not optimize a solution for the user, but simulates the effects of the user modification or strategy.

TPF OPERATIONAL TEST PROCEDURES

The operation of the TPF model system can be verified by running an individual course stored in a permanent data set, and comparing the results against previously obtained results. For purposes of this test, data for course CDP 0129 are stored in a permanent data set called VFY DATA. To use these data for the test run and invoke the TPF model, enter:

TPF VERIFY

```
          TPF MASTER

INPUT THIS RUN:
MASTER
ENTER CARRIAGE RETURN TO CONTINUE

TPF MODEL - WANT INSTRUCTIONS (Y OR N)
_N

PRESENT RUN CONDITIONS
REPORTS:  CTR
INPUT = 1Ø      FYR    = 1976      LYR    = 1976      AOB    = Ø365

ENTER RUN, QUIT, OR FIELD TO BE CHANGED

ENTER REQUEST:
_NOSHO .8ØØ

ENTER REQUEST:
_LEN .9ØØ

ENTER REQUEST:
_SHOW

PRESENT RUN CONDITIONS
NOSHO = .8ØØ      LEN    = .9ØØ
REPORTS:  CTR
INPUT = 1Ø      FYR    = 1976      LYR    = 1976      AOB    = Ø365

ENTER REQUEST:
_RUN
++++END TPF EXECUTION
ENTER: CTRRUN FOR CENTER SUMMARY; CRSRUN FOR COURSE OUTPUTS


13.3Ø.51 CTRRUN
STANDARD FILE - 21 LINES
  2 PAGES IN FILE

+ENTER PAGE NUMBER
1
PAGE   1
                    TPF MODEL -- FISCAL YEAR 1976 SUMMARY
                   ** TOTALS FOR  FLEET TRAINING CENTER **

              SUMMARY (RUN Ø81275)

            ! 1 QR FY 76 ! 2 QR FY 76 ! 3 QR FY 76 ! 4 QR FY 76 ! ANNUAL TOTAL
------------|------------|------------|------------|------------|-------------
# CONVENINGS!   759!      !   695!      !   678!      !   72Ø!      !   2852 !
CRS CAPACITY!239Ø8!      !22514!      !21648!      !24265!      ! 92335 !
UTILIZATION !16681! 69.8%!12959! 57.6%!12193! 56.3%!13333! 54.9%! 55166 ! 59.7%
NO SHOWS    ! 1933! 1Ø.4%! 1581! 1Ø.9%! 1488! 1Ø.9%! 1667! 11.1%!  6669 ! 1Ø.8%
BAKLOG #/WKS! 345Ø! 9 WKS! 2687!11 WKS! 2341!13 WKS! 2188!15 WKS!  2188 !15 WKS
STU. DAYS/AOB!87759! 961.7!69833! 765.3!677Ø2! 741.9!69729! 764.2!295Ø23 ! 8Ø8.3
            !      !      !      !      !      !      !      !      !       !
TOTAL PASS  !16136! 96.7%!12457! 96.1%!11699! 95.9%!12799! 96.Ø%! 53Ø91 ! 96.2%
TOTAL FAIL  !  433! 2.6%!  385! 3.Ø%!  376! 3.1%!  4Ø7! 3.1%!  16Ø1 !  2.9%
SETBACK ACAD!   36! Ø.2%!   27! Ø.2%!   27! Ø.2%!   3Ø! Ø.2%!   12Ø !  Ø.2%
DISENROLL NA!  112! Ø.7%!  117! Ø.9%!  118! 1.Ø%!  127! 1.Ø%!  474 !  Ø.9%
-------------------------------------------------------------------------------
PAGE   1  +PAUSING- ENTER NEXT PAGE NO.
```

FIGURE III-5 SAMPLE PROBLEM: REDUCE NO SHOWS AND COURSE LENGTHS

TPF MASTER

```
INPUT THIS RUN:
MASTER
ENTER CARRIAGE RETURN TO CONTINUE

TPF MODEL - WANT INSTRUCTIONS (Y OR N)
_N

PRESENT RUN CONDITIONS
NOSHO = .8ØØ      LEN   = .9ØØ
REPORTS:  CTR
INPUT = 1Ø      FYR   = 1976      LYR   = 1976      AOB   = Ø365

ENTER RUN, QUIT, OR FIELD TO BE CHANGED

ENTER REQUEST:
_LEN DEACT

ENTER REQUEST:
_CONV .9ØØ

ENTER REQUEST:
_SHOW

PRESENT RUN CONDITIONS
NOSHO = .8ØØ      CONV  = .9ØØ
REPORTS:  CTR
INPUT = 1Ø      FYR   = 1976      LYR   = 1976      AOB   = Ø365

ENTER REQUEST:
_RUN
++++END TPF EXECUTION
ENTER: CTRRUN FOR CENTER SUMMARY; CRSRUN FOR COURSE OUTPUTS

13.35.16 CTRRUN
STANDARD FILE - 21 LINES
  2 PAGES IN FILE

+ENTER PAGE NUMBER
1
PAGE   1
                    TPF MODEL -- FISCAL YEAR 1976 SUMMARY
                  ** TOTALS FOR  FLEET TRAINING CENTER **

              SUMMARY (RUN Ø81275)

             ! 1 QR FY 76 ! 2 QR FY 76 ! 3 QR FY 76 ! 4 QR FY 76 ! ANNUAL TOTAL
-------------!------------!------------!------------!------------!-------------
# CONVENINGS!   694!       !   622!       !   6Ø6!       !   65Ø!       !   2572 !
CRS CAPACITY!218Ø6!       !2ØØ49!       !19794!       !216Ø8!       !  83257 !
UTILIZATION !15955! 73.2%!12455! 62.1%!12192! 61.6%!12934! 59.9%!  53536 ! 64.3%
NO SHOWS    ! 1838! 1Ø.3%! 1491! 1Ø.7%! 1487! 1Ø.9%! 1565! 1Ø.8%!   6381 ! 1Ø.6%
BAKLOG #/WKS! 4431!11 WKS! 4Ø44!13 WKS! 3945!14 WKS! 4Ø74!16 WKS!   4Ø74 !16 WKS
STU DAYS/AOB!9Ø425! 991.Ø!74Ø52! 811.5!73445! 8Ø4.9!7335Ø! 8Ø3.8!311272 ! 852.8
            !      !       !      !       !      !       !      !       !       !
TOTAL PASS  !15443! 96.8%!11959! 96.Ø%!117Ø1! 96.Ø%!12413! 96.Ø%!  51516 ! 96.2%
TOTAL FAIL  !  4Ø2! 2.5%!  382! 3.1%!  374! 3.1%!  398! 3.1%!   1556 !  2.9%
SETBACK ACAD!   32! Ø.2%!   27! Ø.2%!   23! Ø.2%!   27! Ø.2%!    1Ø9 !  Ø.2%
DISENROLL MA!  11Ø! Ø.7%!  114! Ø.9%!  117! 1.Ø%!  123! 1.Ø%!    464 !  Ø.9%
-------------------------------------------------------------------------------
PAGE   1  +PAUSING- ENTER NEXT PAGE NO.
```

FIGURE III-6 SAMPLE PROBLEM: REDUCE NO SHOWS AND COURSE CONVENINGS

This will start the procedure, using VFY DATA as the Master data base. the first message asks the user if he wants instructions. The response should be 'Y', and the display should be as in Figure III-7. Following the display of the TPF instructions and the fields, the present run conditions are displayed. If the run conditions are not exactly as they appear in Figure III-7, they should be modified using the procedure outlined in the TPF Operational Procedures. Basically, this involves deactivating fields that are set, but not desired, by entering:

FIELDNAME DEACT

and by setting fields to the desired value by entering:

FIELDNAME VALUE

Enter SHOW prior to running the model to verify run conditions. Once the run conditions are the same as those shown in Figure III-7, the command RUN should be entered. This entry will be followed shortly by the system display:

END TPF EXECUTION
ENTER: CTRRUN FOR CENTER FORMAT: CRSRUN FOR COURSE FORMAT

At this time, the user should enter the command CRSRUN, and will receive a response:

STANDARD FILE - 25 LINES
1 PAGES IN FILE

+ ENTER PAGE NUMBER

Following this display, enter a carriage return, which should produce the output as shown in Figure III-8. The message:

PAGE 1 +PAUSING - ENTER NEXT PAGE NO.

should be the last line on the display. After comparing the data for accuracy, enter DONE to terminate the display.

The school and center outputs should be verified next. Enter CTRRUN to call the school and center reports. The message:

STANDARD FILE - 21 LINES
3 PAGES IN FILE

should be displayed. Enter a carriage return to display the school format which should be the same as in Figure III-9. Following verification of the school data, enter another carriage return to display the center summary. Again, following verification of this display with Figure III-9, enter another carriage return and verify that the message:

END OF TPF OUTPUT 1 COURSES PROCESSED

is displayed. Enter DONE to terminate this system test.

```
        TPF VERIFY

INPUT THIS RUN:
VERIFY
ENTER CARRIAGE RETURN TO CONTINUE

TPF MODEL - WANT INSTRUCTIONS (Y OR N)
_Y

ENTER: FNAME XXXX  (SET FNAME TO VALUE)
       FNAME DEACT (DEACT FNAME)
       SHOW        (SHOW RUN COND)
       RUN         (RUN MODEL)

FIELDS:  FYR   (NNNN)  LYR   (NNNN)  AOB   (NNNN)  CDP   (AAAA)  SCH   (AAAA)
         FAIL  (NNNN)  NOSHO (NNNN)  DMD   (NNNN)  TREND (NNNN)  CONV  (NNNN)
         LEN   (NNNN)  CAP   (NNNN)  SETB  (NNNN)

REPORTS: CRS   (N)     SCHL  (N)     CTR (N)       PRINT (N)

PRESENT RUN CONDITIONS
REPORTS:  CRS   SCHL  CTR
INPUT = 1Ø     FYR   = 1976     LYR   = 1976     AOB   = Ø365

ENTER RUN, QUIT, OR FIELD TO BE CHANGED

ENTER REQUEST:
_RUN
+END TPF EXECUTION
ENTER: CTRRUN FOR CENTER SUMMARY; CRSRUN FOR COURSE OUTPUTS
```

FIGURE III-7 TPF TEST RUN PROCEDURE

```
        CRSRUN
STANDARD FILE - 25 LINES
  1 PAGES IN FILE

+ENTER PAGE NUMBER

PAGE   1
Ø129      MOTION PICT OPER      --SCHEDULE DATA--    FY   FY+1   FY+2   CRS BACKLOG
                                ANNUAL CONVENINGS    51     Ø      Ø     Ø STU   122
SCHL JNO  CATALOG NO   NEC     COURSE LENGTH DYS   5.Ø   Ø.Ø    Ø.Ø     Ø WKS     6
ELEC Ø162 A-69Ø-ØØ1Ø           COURSE CAPACITY      2Ø     Ø      Ø
                                BUPERS CAPACITY      Ø     Ø      Ø     FY+1 WK   29
P AOB   PFAIL   TYPE    PRTY   ANNUAL DEMAND       48Ø    53Ø    53Ø    FY+2 WK    Ø
  7.Ø   Ø.ØØ    F1      2Ø     BUPERS DEMAND        Ø      Ø      Ø     OFFSET     Ø
                                                     *

                              QUARTERLY PROJECTIONS
            ! 1 QR FY 76 ! 2 QR FY 76 ! 3 QR FY 76 ! 4 QR FY 76 ! ANNUAL TOTAL
------------!------------!------------!------------!------------!-------------
Ø CONVENINGS!   14 !       !   12 !       !    1 !       !    Ø !       !    27 !
CRS CAPACITY!  28Ø !       !  24Ø !       !   2Ø !       !    Ø !       !   54Ø !
UTILIZATION !  217 !  77.5%!   96 !  4Ø.Ø%!    7 !  35.Ø%!    Ø !   Ø.Ø%!   32Ø !  59.3%
NO SHOWS    !   37 !  14.6%!   17 !  15.Ø%!    2 !  22.2%!    Ø !   Ø.Ø%!    56 !  14.9%
BAKLOG Ø/WKS!    Ø !   Ø WKS!    Ø !   Ø WKS!    Ø !   Ø WKS!    Ø !   Ø WKS!     Ø !   Ø WKS
STU DAYS/AOB! 1Ø85 !  11.9 !  48Ø !   5.3 !   35 !   Ø.4 !    Ø !   Ø.Ø !  16ØØ !   4.4
            !      !       !      !       !      !       !      !       !       !
TOTAL PASS  !  217 ! 1ØØ.Ø%!   96 ! 1ØØ.Ø%!    7 ! 1ØØ.Ø%!    Ø !   Ø.Ø%!   32Ø ! 1ØØ.Ø%
TOTAL FAIL  !    Ø !   Ø.Ø%!    Ø !   Ø.Ø%!    Ø !   Ø.Ø%!    Ø !   Ø.Ø%!     Ø !   Ø.Ø%
SETBACK ACAD!    Ø !   Ø.Ø%!    Ø !   Ø.Ø%!    Ø !   Ø.Ø%!    Ø !   Ø.Ø%!     Ø !   Ø.Ø%
DISENROLL NA!    Ø !   Ø.Ø%!    Ø !   Ø.Ø%!    Ø !   Ø.Ø%!    Ø !   Ø.Ø%!     Ø !   Ø.Ø%
-------------------------------------------------------------------------------
PAGE   1  +PAUSING- ENTER NEXT PAGE NO.
DONE
+DISPLAY TPF ENDED
CSS.211 Ø5/Ø7/75

13.38.57
```

FIGURE III-8 TPF TEST RUN - COURSE OUTPUT

```
          CTRRUN
STANDARD FILE - 21 LINES
  3 PAGES IN FILE

+ENTER PAGE NUMBER

PAGE   1
                        TPF MODEL -- FISCAL YEAR 1976 SUMMARY
                                SCHOOL CODE  ELEC

ELEC  SCHOOL SUMMARY (RUN #81275)

             ! 1 QR FY 76 ! 2 QR FY 76 ! 3 QR FY 76 ! 4 QR FY 76 ! ANNUAL TOTAL
-------------!------------!------------!------------!------------!--------------
# CONVENINGS!    14!        !    12!        !     1!        !     #!        !    27 !
CRS CAPACITY!   28#!        !   24#!        !    2#!        !     #!        !   54# !
UTILIZATION !   217!  77.5%!    96!  4#.#%!     7!  35.#%!     #!  #.#%!   32# ! 59.3
NO SHOWS    !    37!  14.6%!    17!  15.#%!     2!  22.2%!     #!  #.#%!    56 ! 14.9
BACLOG #/WKS!     #! # WKS!     #! # WKS!     #! # WKS!     #! # WKS!     # ! # WKS
STU DAYS/AOB!  1#65!   11.9!   48#!    5.3!    35!    #.4!     #!    #.#!  16## !   4.4
             !      !        !      !        !      !        !     !        !      !
TOTAL PASS   !   217!1##.#%!    96!1##.#%!     7!1##.#%!     #!  #.#%!   32# !1##.#%
TOTAL FAIL   !     #!  #.#%!     #!  #.#%!     #!  #.#%!     #!  #.#%!     # !  #.#%
SETBACK ACAD!     #!  #.#%!     #!  #.#%!     #!  #.#%!     #!  #.#%!     # !  #.#%
DISENROLL NA!     #!  #.#%!     #!  #.#%!     #!  #.#%!     #!  #.#%!     # !  #.#%
-------------------------------------------------------------------------------
PAGE   1  +PAUSING- ENTER NEXT PAGE NO.
2
PAGE   2
                        TPF MODEL -- FISCAL YEAR 1976 SUMMARY
                       ** TOTALS FOR TRAPAC TRAINING CENTER **

CENTER CODE N SUMMARY  (RUN #81275)

             ! 1 QR FY 76 ! 2 QR FY 76 ! 3 QR FY 76 ! 4 QR FY 76 ! ANNUAL TOTAL
-------------!------------!------------!------------!------------!--------------
# CONVENINGS!    14!        !    12!        !     1!        !     #!        !    27 !
CRS CAPACITY!   28#!        !   24#!        !    2#!        !     #!        !   54# !
UTILIZATION !   217!  77.5%!    96!  4#.#%!     7!  35.#%!     #!  #.#%!   32# ! 59.3%
NO SHOWS    !    37!  14.6%!    17!  15.#%!     2!  22.2%!     #!  #.#%!    56 ! 14.9%
BACLOG #/WKS!     #! # WKS!     #! # WKS!     #! # WKS!     #! # WKS!     # ! # WKS
STU DAYS/AOB!  1#65!   11.9!   48#!    5.3!    35!    #.4!     #!    #.#!  16## !   4.4
             !      !        !      !        !      !        !     !        !      !
TOTAL PASS   !   217!1##.#%!    96!1##.#%!     7!1##.#%!     #!  #.#%!   32# !1##.#%
TOTAL FAIL   !     #!  #.#%!     #!  #.#%!     #!  #.#%!     #!  #.#%!     # !  #.#%
SETBACK ACAD!     #!  #.#%!     #!  #.#%!     #!  #.#%!     #!  #.#%!     # !  #.#%
DISENROLL NA!     #!  #.#%!     #!  #.#%!     #!  #.#%!     #!  #.#%!     # !  #.#%
-------------------------------------------------------------------------------
PAGE   2  +PAUSING- ENTER NEXT PAGE NO.
3
PAGE   3
END OF TPF OUTPUT    1 COURSES PROCESSED
PAGE   3  +PAUSING- ENTER NEXT PAGE NO.
DONE
+DISPLAY TPF ENDED
CSS.211 #5/#7/75

13.43.32
```

FIGURE III-9 TPF TEST RUN - SCHOOL AND CENTER OUTPUT

SECTION IV

SYSTEMS CAPABILITIES/REQUIREMENTS AND RESOURCES MODEL

MODEL DESCRIPTION AND USER'S GUIDE

SCRR MODEL ARCHITECTURE

The objective of the Systems Capabilities/Requirements and Resources (SCRR) model is to provide Navy training complex officials with sufficient physical resource information (i.e., courses, instructors, classrooms, laboratories) and related analyses to:

a. Assess the feasibility of meeting annual training requirements at the training complex level.

b. Evaluate alternative plans for meeting both short- and long-term training requirements.

c. Assess the utilization of existing resources in the daily operation of the training complex.

The SCRR model will derive the data required to fulfill these objectives through the solution and subsequent sensitivity analysis of the following linear programming problem.

> Determine the maximum student throughput based on an optimal mix of course convenings which a training complex can achieve in a specified period of time, subject to either existing or projected physical resource constraints.

Specifically, a training complex official will be able to use the SCRR model to analyze the projected impact of modifications to training demand or resource availability on student throughput, course convenings, and resource utilization.

The following are presented as examples of modifications which the SCRR model will evaluate.

a. Courses can be added to or deleted from the training complex schedule.
b. Course lengths can be increased or decreased.
c. Course convening frequencies can be altered.
d. Normal course capacities can be increased or decreased.
e. Course utilization percentages can be changed.
f. Student/instructor ratios can be modified.
g. Instructors can be added or deleted.
h. Instructor qualifications can be modified.
i. Instructor availability can be increased or decreased.
j. Classroom and laboratory availabilities can be increased or decreased.

Although training system resource optimization is the primary goal of the SCRR model, the model can also be used to calculate the quantity and mix of resources required to achieve either the current or a user-specified course convening schedule. The SCRR model operating mode can be changed from optimization to calculation by executing the program FIXCONV prior to model execution.

The SCRR model interfaces only with the Master data base. The model user can optionally select data from any organizational element, i.e., a division, department, or activity, for model input. All SCRR model input is then automatically generated based on the contents of the Master data base at the time of model execution.

All SCRR model programs are designed to be interactively executed using telecommunication terminals.

MODEL DESCRIPTION. Figure IV-1 depicts the SCRR model functional flow. The components of the functional flow can be divided into three distinct categories:

a. Data Base

b. SCRR Model

c. User Analysis

The SCRR model input data is generated based on the data contents of the Master data base. After an initial model execution which creates the SCRR input data matrix, the model user may elect to modify the model input data to represent projected training requirement/resource relationships, or to answer "what if" questions without destroying the data stored in the DOTS Master data base. Although the data base is not an integral part of the SCRR Model, model operation is dependent upon the data base.

The five primary components of the SCRR model are:

a. Formulate LP objective function and constraint equations.

b. Compute linear programming solution.

c. Perform post-optimal sensitivity analysis.

d. Format optimal solution and sensitivity analysis output.

e. Modify resource requirements and/or availabilities.

The remaining functional flow components fall into the user analysis category. The user must analyze the optimal solution and the sensitivity analysis results to determine if that solution can be implemented. If not, then he must consider the previous results together with information from other sources, including data from the Training Process Flow (TPF) and the Individualized Training System Simulation (ITSS) models, to modify the resource mix, training requirements, or other model parameters.

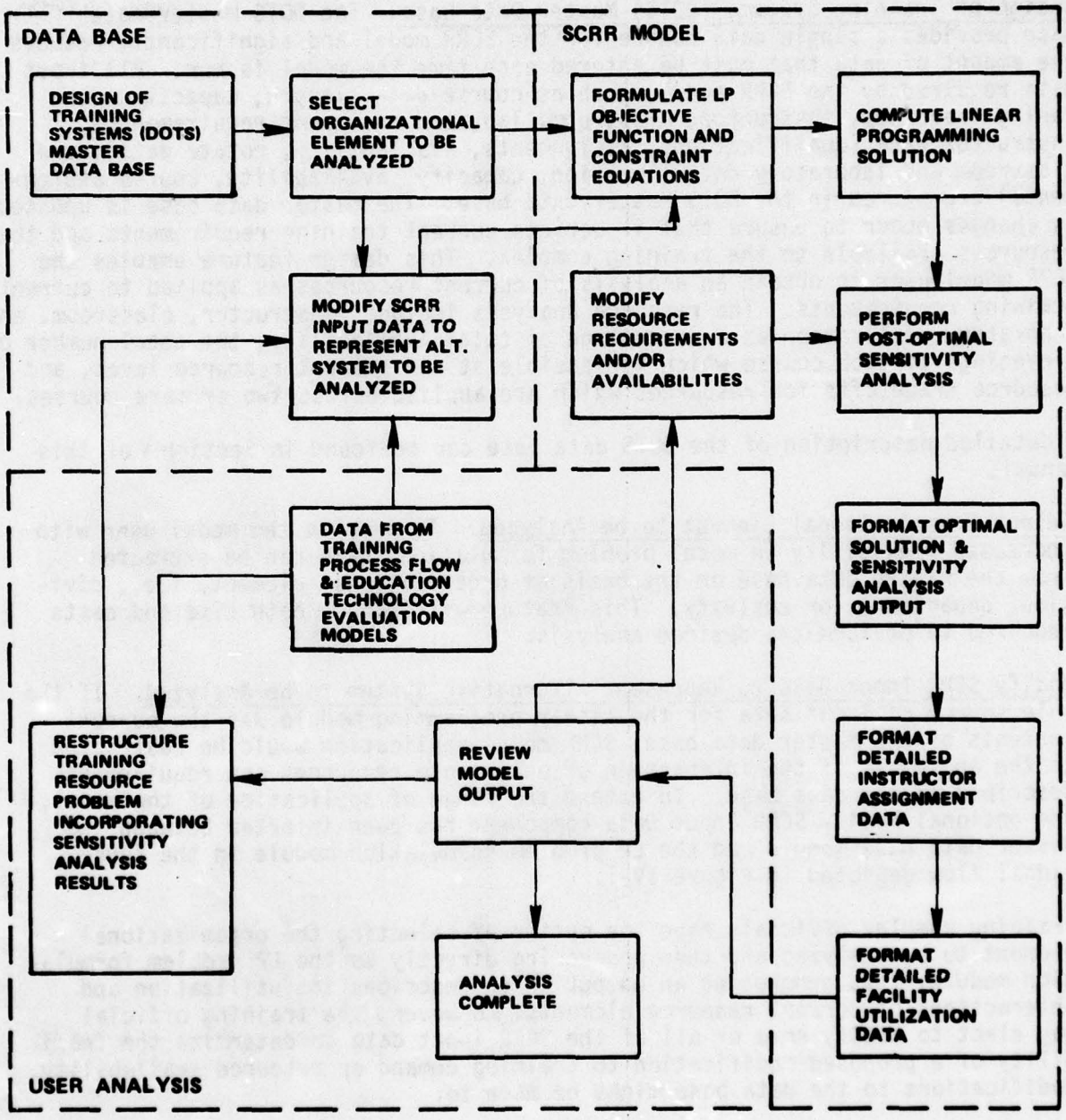


FIGURE IV-1 SCRR MODEL FUNCTIONAL FLOW

Design of Training Systems (DOTS) Master Data Base. The DOTS Master data base provides a single data source for the SCRR model and significantly reduces the amount of data that must be entered each time the model is run. All input data required by the SCRR model, such as course data (length, capacity, convening frequency, instructor, classroom, lab, and equipment requirements), instructor data (qualifications, assignments, availability, rotate date), and classroom and laboratory data (location, capacity, availability, course assignments) are stored in the DOTS Master data base. The Master data base is updated as changes occur to ensure that it depicts current training requirements and the resources available to the training complex. This design feature enables the SCRR model user to obtain an analysis of current resources as applied to current training requirements. The resource analysis includes instructor, classroom, and laboratory utilization as a percentage of total availability, the total number of convenings of each course which is feasible at the current resource level, and resource trade-offs for resources which are applied across two or more courses.

A detailed description of the DOTS data base can be found in Section V of this manual.

Select Organizational Element to be Analyzed. To provide the model user with increased flexibility in model problem formulation, data can be extracted from the Master data base on the basis of organizational element, i.e., division, department, or activity. This feature will reduce both time and costs required to perform the desired analysis.

Modify SCRR Input Data to Represent Alternative System to be Analyzed. If the sole source of input data for the linear programming module was the current contents of the Master data base, SCRR model application would be restricted to the analaysis of the interaction of only those resources and requirements described in the data base. To extend the range of application of the model, the optional Modify SCRR Input Data component has been inserted between the Master data base module and the LP problem formulation module in the functional flow depicted in Figure IV-1.

Training complex officials have the option of selecting the organizational element to be analyzed and then proceeding directly to the LP problem formulation module, thus generating an output which describes the utilization and interactions of current resource elements. However, the training official may elect to modify some or all of the SCRR input data to determine the feasibility of a proposed modification to training demand or resource availability. Modifications to the data base might be made to:

a. Incorporate data from either the TPF or the ITSS models.

b. Incorporate results of previous SCRR model runs.

c. Answer "what if" questions posed by training complex, functional command, or EDTRACOM officials.

Formulate LP Objective Function and Constraint Equations. Linear programming deals with the problems arising out of the need to allocate limited resources among competing activities to meet desired objectives. These problems are characterized by the large number of solutions that satisfy the basic conditions of each problem. The selection of a particular solution as the best solution to a problem depends on some aim or overall objective that is implied in the statement of the problem. A solution that satisfies both the conditions of the problem and the given objective is termed an optimum solution. The complete mathematical statement of a linear programming problem includes a set of simultaneous linear equations or inequalities which represent the conditions of the problem, and a linear function which expresses the objective of the problem.

The linear programming term "objective function" represents the target of the linear programming solution. The answer to the problem must satisfy this requirement. The objective function must be clearly stated and expressed as a linear mathematical function. The stated objective of the SCRR linear programming model is to maximize the student throughput which a school or training complex can achieve in a specified time period subject to existing physical resource constraints. This objective can be expressed mathematically as:

$$\text{MAXIMIZE} \quad \sum_{j=1}^{J} C_j X_j$$

which is shorthand notation for saying

"MAXIMIZE $C_1X_1 + C_2X_2 + C_3X_3 + \ldots + C_JX_J$".

where

X_j = number of annual convenings of course j.
C_j = average class size of course j based upon the normal class capacity and the historical course utilization percentage.

The LP problem formulation module constructs the objective function based on average class size and reformats the function to satisfy the input requirements of the LP module.

Several conditions or contraints must be considered in the formulation of the SCRR linear programming problem.

Course syllabi require a specific amount of classroom instruction time from one or more instructors for each course convening. However, only a limited number of instructors is qualified to teach each course. Therefore, the total amount of time the instructors have available for classroom instruction limits the number of times the course can be convened. The product of the classroom instruction time and the number of convenings cannot exceed the total amount of time available from all qualified instructors.

Each course syllabus also requires that classroom and/or laboratory space be available for each convening. The amount of time each facility is available also represents a limitation to the number of times a course can be convened.

In addition to the constraints imposed by instructors, classrooms, and laboratories, most courses are further restricted in that they must be convened some minimum number of times to fulfill a training requirement to train a minimum number of individuals.

These conditions can be mathematically stated by a set of simultaneous inequalities as follows:

(1) $\sum_{j=1}^{J} a_{ij} X_j \leq TCH_i \quad i = 1, 2, \ldots I.$

where

a_{ij} = number of instructor contact hours required by course syllabus for the ith instructor for one convening of course j.

TCH_i = total contact hours available per year for the ith instructor.

(2) $\sum_{j=1}^{J} b_{kj} X_j \leq TLA_k \quad k = 1, 2, \ldots K.$

where

b_{kj} = number of hours of usage required from laboratory type k for each convening of course j.

TLA_k = total hours per year which laboratory type k is available for use.

(3) $\sum_{j=1}^{J} d_{mj} X_j \leq CRA_m \quad m = 1, 2, \ldots M.$

where

d_{mj} = number of hours of usage required from the type m classroom for one convening of course j.

CRA_m = total hours per year the type m classroom is available for use.

(4) $X_j > MIN_j.$

where

MIN_j = minimum number of convenings per year for course j to meet a minimum training requirement.

The formulation module constructs the entire set of simultaneous inequalities described above, and reformats the constraints to conform to the input requirements of the LP module. The LP input data are compiled based solely on the contents of the Master data base. The model user's only responsibility is to ensure that the system to be analyzed is accurately described in the Master data base. Once SCRR model execution has been initiated, no further user intervention is required.

Compute Linear Programming Solution. The linear programming module accepts formatted input from the LP problem formulation module and calculates either the maximum student throughput and maximum convenings of each course possible per year subject to the expressed physical resource constraints, or the quantity of resources required to attain a specified throughput or convening schedule. Since many excellent texts which provide detailed methodology for linear programming computational procedures are available, these procedures and associated mathematical proofs will not be included in this report. An extensive linear programming bibliography can be found in Dantzig.[1]

Mathematical Programming System III Online System, MPS III/OL,[2] a product developed by Management Science Systems based on IBM's MPS/360, and adapted for interactive use by National CSS, Inc. is utilized to perform the linear programming computations. The MPS III/OL linear programming procedures use the bounded variable/product form of the inverse/revised simplex method.

Perform Post-Optimal Sensitivity Analysis. The purpose of the sensitivity analysis module is to provide information concerning the range of operations in the neighborhood of the optimum solution as calculated by the LP module. The sensitivity analysis will provide information relative to how changing instructor, classroom, or laboratory utilization affects the optimal student throughput, and the range of instructor, classroom, and laboratory utilization hours for which the solution, as originally stated, remains optimal. The sensitivity analysis will also generate information describing the effect of class size on the optimal solution as well as the feasible range of annual convenings, and the effect of changing the number of convenings on the optimal student throughput. The information obtained from the sensitivity analysis should prove to be as valuable as the specification of the optimum solution itself.

There are several reasons for performing a sensitivity analysis. Stability of the optimal solution under changes of parameters may be critical. For example, using the old optimum solution point, a slight variation in the required number of convenings or in instructor requirements may result in a large unfavorable difference in the objective function (student throughput), while a large variation in either of the parameters in another direction may result in only a small difference. The training center official may find it desirable to move away from the optimum solution when variables such as course demand, which are not considered in the SCRR model, are taken into account.

---

[1]G. B. Dantzig, Linear Programming and Extensions. Princeton, N. J.: Princeton University Press, 1963.

[2]National CSS, Inc., MPS III/OL Reference Guide. NCSS Form 978, February 1974.

Instructor, classroom, laboratory requirements and availabilities, minimum convening frequencies, and class capacities are to some extent controllable, and it would be advantageous to know the effects which would result from changing the values of these parameters. Determination of the range of values for each of the parameters for which the solution remains optimum will also identify those parameters to which the optimum solution is extremely sensitive.

Format Optimal Solution and Sensitivity Analysis Output. From the operational point of view, the key component of any model is the output module. Model output represents the primary interface between the model and its user. The SCRR model output is divided into two segments:

a. LP Optimum Solution and Sensitivity Analysis - Resource Data

b. LP Optimal Solution and Sensitivity Analysis - Course Data.

LP Optimal Solution and Sensitivity Analysis - Resource Data. The LP module output and the sensitivity analysis module output have been combined into a single output format for all resource data. Instructor resources are identified by instructor name. Classrooms and laboratories are identified by building and room number. The resource data output section lists the following for each resource:

a. Annual non-classroom hours (instructors only).

b. Annual availability classroom contact hours.

c. Annual requirement classroom contact hours.

d. Requirement vs. availability ratio.

e. Upper and lower limits of resource utilization hours.

f. Student throughput change per unit resource change.

g. Identification of variable which limits utilization range, and whose value will change as the resource level is modified.

LP Optimal Solution and Sensitivity Analysis - Course Data. As with the resource data, the output from both the LP module and the sensitivity analysis module is presented in a single report format for course data. Courses are identified by CDP numbers. The course data section lists the following for each course:

a. Maximum number of annual convenings.

b. Current number of scheduled convenings.

c. Average class size.

d. Range of annual convenings.

e. Student throughput change per course convening change.

f. Range of class sizes to which indicated solution can be applied.

g. Identification of variable which limits convenings range, and whose value will change as the convening level is modified.

Format Detailed Instructor Assignment Data. The LP resource data output shows total annual instructor requirements for each instructor. The instructor assignment listing provides the model user with a detailed breakdown of the total contact hour requirements. The instructor assignment listing contains the following information for each instructor:

a. Course CDP Number for each course contributing to the total contact hour requirement.

b. Average class size.

c. Number of convenings per year.

d. Number of contact hours per course convening.

e. Total annual contact hours required per course.

f. Total annual contact hour requirement per instructor.

g. Total annual availability per instructor.

Format Detailed Facility Utilization Data. The LP resource data output also shows the total annual facility requirements for each building/room combination. The facility utilization listing provides the model user with a detailed breakdown of the total facility requirements. The facility utilization listing contains the following information for each building/room:

a. Course CDP number for each course contributing to the total requirement.

b. Average class size.

c. Number of convenings per year.

d. Number of hours required per course convening.

e. Total annual hours required per course.

f. Total annual hours required per facility.

g. Annual availability per facility.

Review Model Output. After the training complex official has analyzed the linear programming optimal solution and the output from the sensitivity analysis, he is finally in a position to interpret and evaluate the model results. To successfully interpret the model results, the official must be familiar with the mathematical formulation of the linear programming problem. The objective of this familiarization requirement is not to understand the internal mathematical manipulations required to achieve the linear programming solution,

but to be aware of simplifications and deviations from reality that were, of necessity, built into the initial linear programming solutions. Model results must be interpreted taking all assumptions and simplifications fully into consideration.

Assuming that the official either accepts the model results or modifies the model output, based on his experience and intuitive feeling for the situation being analyzed, his next task is to evaluate the results in terms of implementation feasibility. If, because of some physical, monetary, or political restriction, full or partial implementation is not feasible, the problem statement must be reformulated utilizing any new data or insights resulting from the initial problem solution and sensitivity analysis.

Upon completion of the model output review, the model user may decide that:

a. All alternatives have been evaluated and the analysis is complete.

b. The original problem should be restructured incorporating results from the current model run.

c. A parametric study of resource requirement factors should be performed.

Modify Resource Requirements and/or Availabilities. Assuming that the basic training system structure (number of courses, instructor assignments and qualifications, student/instructor ratios, etc.) described in the Master data base is feasible, the model user may elect to perform a parametric analysis of the resource requirement and availability variables, to investigate resource tradeoffs. A parametric analysis is a systematic study of changes in resource parameter values in the linear programming model. Resource parameters capable of being modified using model-related programs as opposed to modifying the SCRR input data are course utilization, class capacity, course length, and resource availability.

Restructure Training Resource Problem Incorporating Sensitivity Analysis Results. When review of SCRR model output indicates that the basis training resource data contained in the Master data base must be restructured to find a feasible solution, the training complex official must decide how to modify the current sytstem structure. This restructuring could take the form of transferring instructors between schools, implementing instructor cross-training programs, or modifying course convening schedules, for example. The system restructure module (See Figure IV-1) completes the interative cycle. Based on SCRR model results, experience, and intuition, an official has the ability to modify initial problem statements or to develop new potential alternative solutions to be evaluated. At this point, the official need only remodify the SCRR input data to initiate an additional cycle of the interative process.

Data From Training Process Flow and Individualized Training System Simulation Models. The results of the TPF and the ITSS models may suggest modifications to several of the SCRR model variables; e.g.:

a. Several courses should be individualized, reducing instructor requirements and calling for modifications to classroom and laboratory space requirements.

b. Student/instructor ratios should be increased for the lab sessions in several courses to reduce failure rates.

c. Convening frequencies should be increased to reduce the wait time required to attend several courses.

d. Convening frequencies should be reduced for low utilization courses.

The SCRR input data can be modified to reflect changes such as those listed above prior to initiating the SCRR model.

MODEL ASSUMPTIONS. A mathematical model of a system is a collection of mathematical relationships which characterize the feasible solutions of the system. By feasible solutions, is meant those solutions which can be carried out under the system's limitations. The technique utilized to solve the SCRR mathematical model is linear programming. Linear programming establishes the optimal system solution by iteratively evaluating feasible solutions against an expressed objective. Before the linear programming technique can be used, several basic requirements must be fulfilled. This section will discuss these basic requirements and demonstrate that these requirements are fulfilled by the SCRR model application. Also included in this section is a discussion of the real world interpretation of a linear programming optimal solution. Assumptions and limitations relative to the SCRR mathematical model variables are also discussed.

Linear Programming Basic Requirements. As the name "linear programming" implies, both the objective function and every constraint function must be linear. Linearity is a primary requirement of linear programming. A linear relationship is essentially defined by two properties, proportionality and additivity. In addition to these two major characteristics, three additional requirements must be adhered to: nonnegativity; divisibility; and deterministic coefficients.

Proportionality. Linearity requires a proportionality or a simple multiplicative relationship between the units of resource requirements and the number of convenings of each course. For example, if six instructor l contact hours are required to convene course A one time, then 12 hours are required for two convenings, and 24 hours are required for four convenings, etc. The amount of resource required is the same for the n-th convening as it is for the first. This is an important property of linearity from the practical point of view. If, for example, it was the case for some instructional curriculum that 60 hours of a given resource were required to attain 30 convenings of some course, but only 100 hours of the resource were required to attain 60 convenings of this course, then the proportionality assumption would not hold. It is obvious, without explanation, that for the SCRR model application, the resource requirements are proportional to the number of course convenings and to the student throughput for a constant class size. The objective function of the current SCRR model formulation is to maximize student throughput based on specified course capacities. Course requirements are calculated based on the same course capacities.

Additivity. The additivity property of linear relationships states that the measures of effect as calculated through the objective function, and the levels of resources as expressed in the constraint equations, must both be additive. The objective function measure of effect is student throughput, which is calcu-

lated by multiplying the course capacity by the number of convenings. Thus, if course A's capacity is 6 and it is convened 10 times each year, the annual throughput for A is 60, and if course B has a capacity of 20 and is convened 15 times per year, its throughput is 300. The additivity property states that the total throughput for the two courses is then 360. A similar example will demonstrate the additivity property's involvement in the constraints of the linear programming model. Instructor 1 teaches both courses A and B. Twenty-five hours of his time are required for each convening of course B. Since A meets ten times each year and B fifteen times, instructor 1's total requirement is 400 hours (250 for A and 150 for B). Like the above example, all SCRR model resource requirements are additive across all courses.

Nonnegativity. The nonnegativity property states that while any positive multiple of course convenings is possible, negative course convenings are not possible. Adherence to this restriction is ensured through the MPSIII/OL program. The MPSIII/OL program allows the user to specify both upper and lower bounds for the decision variables. If a lower bound is not specified, the program assumes a value of zero. Unspecified upper bounds are set equal to infinity. In the case of the SCRR model application, the LP problem formulation algorithm sets the lower bound for course convenings equal to the number of annual convenings (which is always a positive quantity) stored in the data base. A negative number of course convenings is meaningless.

Divisibility. The divisibility property requires that fractional levels of the decision variables be permissable. In many linear programming models, the decision variables would have physical significance only if they have integer values. There is no guarantee that the solution procedure utilized within the SCRR model will yield an integer solution. If an integer solution is required, the common procedure is to round the non-integer optimal solution down to the nearest integer. This course of action could produce two problems. First, this integer solution need not be feasible. Second, even if it is feasible, this solution need not be too near optimality. Since fractional values for the number of course convenings is interpretable, the SCRR model application fulfills the divisibility property requirement. A fractional course convening is interpreted as a course which is convened but is not completed in a calendar year. For example, 0.5 convenings could be associated with a two week course which is convened the last week of a calendar year.

Deterministic Coefficients. All of the coefficients in a linear programming model are assumed to be known constants. In the SCRR model application, this includes class capacities, instructor and space requirements per convening, and instructor and space availabilities. The fact that any or all of the LP coefficients may not be known constants does not invalidate model results, but does require the expenditure of additional effort. (A sensitivity analysis is generally performed to determine the effect on the optimal solution if particular parameters take on other possible values.) Sensitivity analysis is employed to determine the effect of changing the value of a single parameter. It is often of interest to investigate making simultaneous changes in a number of parameters and to study what happens as the magnitude of these simultaneous changes increase. A systematic study of such changes in certain parameters of a linear programming model is the objective of parametric linear programming. A post optimal sensitivity analysis is built into the SCRR model. Sensitivity analysis results are included in the SCRR model output. Parametric programming may be performed by the model user by systematically varying the parameters of interest, rerunning the SCRR model, and comparing results.

Interpretation of LP Optimal Solution. The SCRR model can be operated in two different modes. The number of course convenings can be specified by the user. The model will then calculate the resources required to achieve that number of convenings, and compare the required resources with present resource capabilities. On the other hand, the SCRR model could be run against the data base which depicts current resource capabilities and specifies a minimum number of course convenings. The model will determine the maximum student throughput which could be attained with current resources. Model results from the former operating mode are straight-forward and require no additional explanation. Interpretation of model results in the latter case is more complex.

Consider the following example. Course A is currently scheduled to convene 24 times per year. The SCRR model is run to optimize throughput based on current resources. Model results indicate that course A should be convened 93.7 times per year. Since the user is aware that course A utilization is currently averaging about 70 percent and a reduction in convening frequency is being considered, the model results appear absurd. However, the objective of the SCRR model was to maximize throughput, which is defined as course capacity times the number of convenings, subject to current resource capabilities. Throughput can be increased only by increasing convening frequency. Convening frequency is limited only by available resources. Capability to increase convening frequency to nearly four times the current schedule implies that present course A resources are being utilized approximately 25 percent of the available time. Therefore, the question the SCRR model user should be considering is not should course A be convened 93 times per year, but how can course A resources be utilized more effectively? Resource availabilities stored in the data base should be examined. Perhaps the original availability estimate was too high. Frequent curricula updates may reduce the time available for classroom instruction. Or the instructor(s) could be cross-trained to instruct one or more additional courses. The same model output which indicated 93.7 convenings for course A, may also have pointed out other courses which could not meet minimum convening requirements because of lack of resources.

The user should investigate all the above possibilities. Model input parameters could be modified and the model rerun to assist in the evaluation of the feasible alternatives.

Assumptions and Limitations of SCRR Model Variables. The laboratory and classroom facilities; i.e., the number of spaces available for course presentation. Lab equipment, training aids, and other major equipment installed in these spaces, are considered to be fixed in their availability in the short run (up to two years), but variable over longer time spans. Therefore, a time lag of from one to two years is assumed between a decision to procure major equipment or to construct classroom or lab facilities, and the completion of the installation or construction.

Although short-range availability of classrooms and laboratories is considered fixed, an estimate of the availability of individual classrooms or labs has not been attempted. A uniform availability of 40 hours per week for 52 weeks per year or a total of 2080 hours per year has been assumed for all classroom and laboratory spaces. The SCRR model user has three options relative to space availability: (1) maintain the assumption of a uniform 2080 hours per year

availability; (2) establish the availability on a room by room basis for the training complex; or (3) utilize the SCRR model to perform parametric studies to determine the effect of facility availability.

The authorized allowance of instructors is considered to be fixed in the short run. The actual on-board count of instructors is considered variable in both the short and the long run. In the short run, variations in the on-board count may be caused by many factors (temporary additional duty, vacations, illness, time lag between assignment rotation and receipt of replacement). In the long run, higher authorities can change the instructor allowance as a function of major changes in curriculum or requirements, changes in command mission, or the general level of manpower authorizations.

On-board instructor count can be easily maintained within the data base. However, given on-board count, the key SCRR model variable becomes instructor availability. Instructor availability is the number of hours per year an instructor has available for classroom instruction. Availability does not include supervisory requirements, military duties, preparation for instruction, duties related to instruction, annual leave, illness, special training, or break-in time. Annual instructor availability is currently projected based on historical activity profile data or, when historical data is not available, an instructor's or supervisor's best estimate. This projection assumes that all instructor activities will be performed in the same ratio throughout the year. However, time expended on several activities such as curriculum development, will fluctuate over a wide range from year-to-year. Actual classroom instruction time, on the other hand, is nearly uniform.

It is assumed that budget does not constrain the SCRR model solution in the short run. However, in the long run, budget constraints of a capital nature may alter the SCRR optimal solution, in that student throughput could be affected by the funding available for new construction and/or procurement of new equipment.

Course curricula are considered fairly inelastic in the short run. Drastic curriculum changes require a considerable amount of time to determine new requirements, develop new material, and secure headquarters review and approval. However, numerous minor changes to courses take place frequently, and a course may be dropped as a result of sustained low utilization.

Student/instructor ratios are generally a function of curriculum requirements. The generally accepted rule used in the establishment of these ratios is that the ratio of trainees per instructor for each instructional situation should be set at that point which yields the highest possible ratio without serious detriment to the quality of instruction.

The optimum ratio should be based on consideration of the type of equipment, safety, and teaching effectiveness for the particular teaching situation. Since no more specific procedures other than the above exist, the establishment of student/instructor ratios remains highly subjective, and should be closely monitored by training complex officials.

The SCRR model calculates all instructor requirements based on historical course utilization. Calculated instructor requirements will be accurate for courses with relatively uniform utilization. Requirements for courses with increasing,

decreasing, or widely fluctuating utilization will be either higher or lower than actual instructor requirements.

INPUT PARAMETERS DESCRIPTION

SCRR model input parameters contained in the DOTS Master data base are described below.

a. CDP Number - The CDP number is a four position alphanumeric field used by NITRAS as a course identifier. Each course is assigned a unique CDP number. All data elements contained in the Course and Facilities files are keyed to the CDP number.

DATA BASE REFERENCE:

FILE	LEVEL	FIELD NAME	FIELD NUMBER	TYPE	LENGTH
COURSE	1	CDP	1	A	4
FACIL	1	CDP	1	A	4

b. Course delete flag - The delete flag is set to one (1) for all active courses. A value of zero (0) indicates that the course has been disestablished.

DATA BASE REFERENCE:

FILE	LEVEL	FIELD NAME	FIELD NUMBER	TYPE	LENGTH
COURSE	1	DEL	3	A	1

c. Number of convenings per year - The number of times each course is scheduled to convene over the next twelve months. Course schedules are based on course capacities and projected training requirements.

DATA BASE REFERENCE:

FILE	LEVEL	FIELD NAME	FIELD NUMBER	TYPE	LENGTH
COURSE	1	CONV1	6	I	3

d. Class capacity - The planned maximum number of students that can attend any one convening of the course. Capacities are based upon training considerations, such as instructor to student ration, availability of training equipment, workshop, laboratories, and mock-up facilities, as well as physical considerations such as classroom size.

DATA BASE REFERENCE:

FILE	LEVEL	FIELD NAME	FIELD NUMBER	TYPE	LENGTH
COURSE	1	CAP1	8	I	3

e. Course utilization - The ratio of the total number of students enrolled in the course over a one year period to the annual class capacity.

DATA BASE REFERENCE:

FILE	LEVEL	FIELD NAME	FIELD NUMBER	TYPE	LENGTH
COURSE	1	UCAP	38	I	3

f. Student/Instructor ratio - A numerical index describing the number of trainees per instructor.

DATA BASE REFERENCE:

FILE	LEVEL	FIELD NAME	FIELD NUMBER	TYPE	LENGTH
COURSE	2	F1	63	I	9

g. Contact hours - The number of instructional contact hours taught at a given ratio of trainees per instructor. A contact hour represents sixty minutes of instruction. This refers to clock hours of curriculum time devoted to actual instruction, exclusive of breaks, administrative time, lunch, medical, dental, etc.

DATA BASE REFERENCE:

FILE	LEVEL	FIELD NAME	FIELD NUMBER	TYPE	LENGTH
COURSE	2	F2	54	I	4

h. Instructor ID - The social security number is used as a unique instructor identification number to facilitate data manipulation.

DATA BASE REFERENCE:

FILE	LEVEL	FIELD NAME	FIELD NUMBER	TYPE	LENGTH
COURSE	2	F1	53	I	9
INST	1	ID	1	I	9

i. Current assignment - A value of one (1) denotes that an instructor is currently assigned to a particular course. An instructor may be assigned to more than one course in situations where course lengths are less than convening frequency. Thus an instructor assigned to a one week course which is convened the first week of every month, could also be assigned to another one week course which is convened the third week of every month.

DATA BASE REFERENCE:

FILE	LEVEL	FIELD NAME	FIELD NUMBER	TYPE	LENGTH
COURSE	2	F3	55	I	1

j. Instructor name - Instructors are identified by the last name in addition to their social security number.

DATA BASE REFERENCE:

FILE	LEVEL	FIELD NAME	FIELD NUMBER	TYPE	LENGTH
COURSE	3 (Virtual)	NME	56	A	16
INST	1	NME	2	A	16

k. Instructor activity profile data - Time is allocated on an annual basis against the following fourteen categories.

CONT - Hours Contact with Students
FLET - Special Fleet Assistance
CDEV - Curriculum Development/Task Analysis
EQMT - Equipment/Training Aid Maintenance
PREP - Preparation for Instruction and Related Duties
MDMT - Military/Physical Training/Other
ADMN - Supervision/Administration
ANLV - Annual Leave
SLIB - Special Liberty
SKLV - Sick Leave
FTRN - Factory Training
ISCH - Formal Instructor/Supervisor School
BKIN - In-service Instructor Training ("break-in" time)
CTRN - In-service Instructor Cross-training Due to Low Student Load

Category names are stored in the data base by the indicated codes; e.g., CONT, FLET, etc.

DATA BASE REFERENCE:

FILE	LEVEL	FIELD NAME	FIELD NUMBER	TYPE	LENGTH
INST	1	See Above	13-26	I	4

l. Building - The name or number of the building in which a classroom or laboratory space is located.

DATA BASE REFERENCE:

FILE	LEVEL	FIELD NAME	FIELD NUMBER	TYPE	LENGTH
FACIL	2	BLDG	3	A	4

m. Room number - The name or number of a classroom or a laboratory space.

DATA BASE REFERENCE:

FILE	LEVEL	FIELD NAME	FIELD NUMBER	TYPE	LENGTH
FACIL	2	RM	4	A	4

n. Required hours - Required hours represent the number of hours the indicated space is required to convene one session of the referenced course.

DATA BASE REFERENCE:

FILE	LEVEL	FIELD NAME	FIELD NUMBER	TYPE	LENGTH
FACIL	2	REQHRS	7	F	6.1

o. Facility available hours - The number of hours, on an annual basis, the space is available for instructional purposes.

DATA BASE REFERENCE:

FILE	LEVEL	FIELD NAME	FIELD NUMBER	TYPE	LENGTH
FACIL	2	AVHRS	8	I	5

OUTPUT PARAMETERS DESCRIPTION

The SCRR model utilizes a linear programming technique to optimize student throughput, subject to limitations of resources required to convene training courses. One of the model output parameters, number of course convenings, provides the basis for the student throughput calculation. The levels of resources required to achieve the optimal student throughput are the only other model output parameters. One section of model output dealing with the LP optimal solution and the sensitivity analysis results is devoted to each type of parameter. The Instructor Assignment Listing is generated to provide the model user with a detailed allocation of the total instructor contact hour requirement. As the name implies, the listing depicts course assignments for

each instructor. Contact hour requirements are detailed for each course assignment. The Facility Utilization Listing is identical to the Instructor listing in format but contains data describing annual facility utilization. Both the course and resource parameters will be described through an explanation of the four SCRR model output listing:

a. Instructor Assignment Listing

b. Facility Utilization Listing

c. LP Optimal Solution and Sensitivity Analysis - Resource Data

d. LP Optimal Solution and Sensitivity Analysis - Course Data

INSTRUCTOR ASSIGNMENT LISTING. A sample printout of the Instructor Assignment listing is presented in Figure IV-2. The listing identifies all courses taught by each instructor. For example, Brooks teaches both 510G and 536K, while Dorsey teaches only 536K. The assignment listing also shows the average class size, the current number of course convenings scheduled, the contact-hour requirement per convening for each instructor, and the total annual contact-hour requirement based on the current number of convenings. For example, course 510G is currently scheduled to convene 6 times this year. The average class size, based on class capacity and historical course utilization, is 7 students. Based on an average class size of 7 students, instructor Brooks is required an average of 20 hours per convening or a total of 120 hours per year for course 510G.

From Figure IV-2, both Paul and Ward also teach course 510G. Both Paul and Ward are required an average of 20 contact hours per 510G convening. The total 510G instructor requirement per convening is 60 hours. If 510G were a two week course (30 hours per week), Brooks, Paul and Ward could alternate teaching the course, each instructor teaching the course 2 times during the year. If 510G were a one week course which required two instructors per convening (60 contact hours per week), each of the three instructors could assist in teaching the course 4 times during the year. In both of the above examples, each of the three instructors taught 510G a total of 120 hours during the year, or an average of 20 hours per convening.

The Instructor Assignment Listing also sums the annual course contact-hour requirements and shows the projected annual availability for each instructor.

FACILITY UTILIZATION LISTING. A sample printout of the Facility Utilization Listing is shown in Figure IV-3. The listing identifies all courses taught in each building/room, the requirement per course convening for each facility, and the total annual requirement based on the current number of convenings.

LP OPTIMAL SOLUTION AND SENSITIVITY ANALYSIS - RESOURCE DATA. The linear programming module calculates the number of course convenings required to maximize student throughput. This portion of the SCRR model output specifies the level of resources required to attain the optimal number of course convenings. The

INSTRUCTOR	COURSE	CLASIZE	CONV	HRS/CONV	TOT HRS	AVAIL HRS
BROOKS	51ØG	7	6	2Ø.Ø	12Ø.Ø	
	536K	1Ø	5Ø	9.3	465.Ø	
				TOT =	585.Ø	1Ø42
BROWN	51ØV	7	12	3Ø.Ø	36Ø.Ø	
	536K	1Ø	5Ø	9.3	465.Ø	
				TOT =	825.Ø	1Ø41
DORSEY	536K	1Ø	5Ø	9.3	465.Ø	1Ø42
HARRISON	51ØW	3	24	18.Ø	432.Ø	1Ø42
HUNT	461V	2	12	42.Ø	5Ø4.Ø	1Ø42
OBERLE	5698	14	6	13Ø.Ø	78Ø.Ø	1ØØ2
PAUL	51ØG	7	6	2Ø.Ø	12Ø.Ø	
	536P	7	12	24.Ø	288.Ø	
				TOT =	4Ø8.Ø	1Ø42
ROHLFING	51Ø7	8	24	2Ø.Ø	48Ø.Ø	1Ø42
SPARKES	51Ø7	8	24	2Ø.Ø	48Ø.Ø	1Ø42
STEWART	51Ø7	8	24	2Ø.Ø	48Ø.Ø	1Ø42
STREIT	51Ø7	8	24	2Ø.Ø	48Ø.Ø	1Ø42
WAGNER	5698	14	6	13Ø.Ø	78Ø.Ø	1Ø42
WARD	51ØG	7	6	2Ø.Ø	12Ø.Ø	
	536P	7	12	24.Ø	288.Ø	
				TOT =	4Ø8.Ø	1Ø42

INSTRUCTOR ASSIGNMENT LISTING

FIGURE IV-2

BLDG / RM	COURSE	CLASIZE	CONV	HRS/CONV	TOT HRS	AVAIL HRS
FLTLHGR	51ØW	3	24	12.Ø	288.Ø	2Ø8Ø
N-3Ø1Ø6	51ØG	7	6	33.Ø	198.Ø	2Ø8Ø
N-3Ø1Ø7	51ØG	7	6	27.Ø	162.Ø	
	5698	14	6	11Ø.Ø	66Ø.Ø	
				TOT =	822.Ø	2Ø8Ø
N-3Ø1Ø8	5698	14	6	4Ø.Ø	24Ø.Ø	2Ø8Ø
N-3Ø167	536K	1Ø	5Ø	2Ø.Ø	1ØØØ.Ø	2Ø8Ø
N-3Ø18Ø	461V	2	12	42.Ø	5Ø4.Ø	
	536P	7	12	48.Ø	576.Ø	
				TOT =	1Ø8Ø.Ø	2Ø8Ø
N-3Ø181	51ØV	7	12	3Ø.Ø	36Ø.Ø	
	51ØW	3	24	6.Ø	144.Ø	
				TOT =	5Ø4.Ø	2Ø8Ø
N19A12Ø	51Ø7	8	24	4Ø.Ø	96Ø.Ø	2Ø8Ø
N19A122	51Ø7	8	24	2Ø.Ø	48Ø.Ø	2Ø8Ø

FACILITY UTILIZATION LISTING

FIGURE IV-3

output specifies the levels to which each individual resource can increase and decrease before it is required to rerun the SCRR model. Also identified is the incremental student throughput change per unit increase or decrease up to those levels. This output assumes that the rest of the problem is unaltered; that is, the remaining input data are left constant, and the solution is adjusted as necessary to maintain feasibility and optimality.

Resource. An example of the LP Optimal Solution and Sensitivity Analysis - Resource Data output is shown in Figure IV-4. The first column identifies the resources required by all courses included in the model run. Instructors are identified by the first eight letters of their last name. Facilities are identified by building and room number.

Annual Non-classroom Hours. Column two summarizes the total hours per year spent out of the classroom. This figure is derived for each instructor by summing the hours for each of the following 13 activities from the instructor activity profile:

FLET - Special Fleet Assistance
CDEV - Curriculum Development/Task Analysis
EQMT - Equipment/Training Aid Maintenance
PREP - Preparation for Instruction and Related Duties
MDMT - Military/Physical Training/Other
ADMN - Supervision/Administration
ANLV - Annual Leave
SLIB - Special Liberty
SKLV - Sick Leave
FTRN - Factory Training
ISCH - Formal Instructor/Supervisor School
BKIN - In-service Instructor Training ("break-in" time)
CTRN - In-service Instructor Cross-training Due to Low Student Load

The instructor activity profile, which contains the hours per year expended on the above activities, is based on historical data when available. When historical data is not available, the activity profile is estimated by either the instructor or supervisory personnel. The above 13 activities plus the 14th activity, instructor contact time, should sum to 2,080 hours per year based on a 40-hour work week (40 hr/wk x 52 wk/yr).

Annual Availability Classroom Contact Hours. Column three in Figure IV-4 denotes the number of hours per year each resource is available to fulfill the specified course convening requirements.

Instructor Availability. The instructor's availability for classroom instruction is also derived from the instructor activity profile data.

Projecting availability based on historical or estimated annual hours assumes that all activities will be performed in approximately the same ratio during the next year.

RESOURCE	ANNUAL NON-CLASSROOM HOURS	ANNUAL AVAIL CLASSROOM CONTACT HOURS	ANNUAL REQMT CLASSROOM CONTACT HOURS	REQMT VS AVAIL RATIO	RESOURCE UTIL. RANGE MINIMUM MAXIMUM	THROUGHPUT CHANGE PER UNIT RESOURCES CHANGE	LIMITING VARIABLE
DORSEY	838	1042	681	65	519.0	-0.688	510V
					681.0	NO CHG	
HUNT	838	1042	718	68	504.0	-0.048	N-30180
					1504.0	-0.098	PAUL
STEWART	838	1042	1040	99	488.0	-0.400	N19A120
					1040.0	NO CHG	
WARD	838	1042	1042	100	880.0	-0.196	PAUL
					1042.0	NO CHG	
BROWN	839	1041	1041	100	879.0	-0.922	HUNT
					1148.0	0.922	461V
PAUL	838	1042	1042	100	880.0	-0.196	HUNT
					1042.0	0.196	WARD
STREIT	838	1042	1040	99	480.0	-0.400	N19A120
					1040.0	NO CHG	
ROHLFING	838	1042	1040	99	480.0	-0.400	N19A120
					1040.0	NO CHG	
BROOKS	838	1042	1042	100	935.0	-0.154	461V
					1204.0	0.154	HUNT
SPARKES	838	1042	1040	99	480.0	-0.400	N19A120
					1040.0	NO CHG	
HARRISON	838	1042	1042	100	432.0	-0.167	510W
					3120.0	0.167	FLTLHGR
OBERLE	878	1002	1002	100	780.0	-0.108	5698
					1042.0	0.108	WAGNER
WAGNER	838	1042	1002	96	780.0	-0.108	OBERLE
					1002.0	NO CHG	
FLTLHGR		2080	695	33	288.0	-0.250	HARRISON
					694.7	NO CHG	
N-30106		2080	596	28	419.1	-0.093	BROOKS
					862.9	-0.417	510V
N-30107		2080	1335	64	1190.7	-0.114	BROOKS
					1553.9	-0.510	510V
N-30108		2080	308	14	240.0	-0.350	OBERLE
					308.3	NO CHG	
N-30167		2080	1465	70	1116.1	-0.320	510V
					1464.5	NO CHG	
N-30180		2080	2080	100	1866.0	-0.048	461V
					2404.0	0.048	HUNT
N-30181		2080	707	34	504.0	-0.500	HARRISON
					869.3	-0.688	510V
N19A120		2080	2080	100	960.0	-0.200	5107
					2084.0	0.200	SPARKES
N19A122		2080	1040	50	480.0	-0.400	N19A120
					1040.0	NO CHG	

LP OPTIMAL SOLUTION AND SENSITIVITY ANALYSIS - RESOURCE DATA

FIGURE IV-4

The model user should be aware of this assumption and should revise availability either upward or downward based on additional information which is not reflected in the activity profiles. For example, instructor B has been involved in curriculum development an average of 20 hours per week for a 6 month period. During that time period he has instructed in the classroom an average of 10 hours per week. The annual availability projection for instructor B would be 500 hours. However, the model user is aware that the curriculum development task is nearing completion and that instructor B will soon be available for classroom instruction 30 hours per week. Instructor B availability should be modified to reflect this information. Thus a more realistic availability might be 1500 hours per year (30 hours/week X 50 weeks).

Facility Availability. Facility availability is stored as a constant in the RAMIS facility file. Current data for most classrooms and laboratories reflect 40 hours per week for 52 weeks or 2080 hours. This figure should be modified by the model user to reflect additional information relative to a specific facility. For example, a trainer may require preventive maintenance to be performed one day every week, reducing its availability to 32 hours per week, or 1664 hours per year.

Annual Requirement Classroom Contact Hours. The annual requirement (Column 4) identifies the total number of hours per year each resource is required to achieve the optimal number of course convenings. (See discussion on maximum annual convenings in course data section.)

Requirement Vs. Availability Ratio. Requirement vs. availability ratio (Column 5) is the ratio of required hours to available hours.

Resource Utilization Range. The resource utilization range (Column 6) indicates the level to which each resource may be increased or decreased without rerunning the SCRR model. Changes in resource level beyond this range will necessitate modifying the data base to reflect the new resource levels, and rerunning the SCRR model. Resource utilization changes within the specified range will affect the value of the optimal solution which is stated in terms of optimal student throughput. The magnitude of this effect (either positive or negative) is indicated by the next column, throughput change per unit resource change.

An example will illustrate the above explanation. Instructor Hunt (from Figure IV-4) utilization range (Column 6) is 504-1504 hours. Hunt's optimum utilization level (the level which maximizes school student throughput) is 718 hours. If Hunt's availability were to decrease below 504 hours, the indicated optimal solution would change. From the Instructor Assignment Listing (Figure IV-2), it is seen that Hunt instructs course 461V. Since minimum instructor requirements for course 461V are 504 hours (42 hours per convening X 12 convenings per year),

a decrease below 504 hours would result in an infeasible solution. Also, Hunt's utilization cannot be extended beyond 1504 hours without changing the optimal solution.

The resource utilization range denotes mathematical feasibility only. Physical limitations are not considered in the LP sensitivity analysis output. Thus the maximum instructor utilization cannot physically exceed 2000 hours (40 hours per week X 50 weeks per year), even though the top end of the utilization range shown in column 6 may be greater.

A utilization decrease in the range 504-718 would decrease the indicated optimum student throughput by 0.048 (Figure IV-4, Column 7) for each hour of decreased utilization. On the other hand, should utilization be increased beyond the 718 hour level, the optimal student throughput would decrease by 0.098 per hour of increased utilization. This relationship between instructor utilization and student throughput will be examined in detail in the following sections.

Throughput Change Per Unit Resource Change. This (Column 7) identifies the change in the optimal student throughput which will result from a one hour change in resource utilization. As long as the resource utilization level remains within the indicated utilization range, the SCRR model need not be rerun. The effect of the resource change can be determined by a simple calculation using the throughput change per unit resource change.

The sensitivity analysis output, in particular the throughput change per unit resource change column, assumes a continuous relationship between the parameter representing the course convenings and the resource requirements parameter. For example, if 10 contact hours are required to convene a course one time, it is assumed that 11 contact hours will result in 1.1 course convenings, 15.5 contact hours will net 1.55 convenings, and 20 contact hours will provide 2 course convenings. Since all convenings are stated in terms of convenings per year, it is possible to interpret a fractional convening as a course which convenes prior to the end of the year but does not conclude until the following year; e.g., 0.5 convenings indicate that a course is one-half complete at the end of the year. However, except for this year-end interpretation, fractional convenings have no meaning in the real world. Once a course is convened, it is always completed. Therefore, in reality, the convening resource relationship, although linear, is not continuous. It is important that the model user keep this fact in mind when interpreting the sensitivity analysis output.

Another example from the LP Optimal Solution and Sensitivity Analysis - Resource Data output (Figure IV-4) will be used to demonstrate the significance of the throughput change per unit resource change. Instructor Hunt's utilization range is indicated to be 504-1504 hours. Hunt's annual availability is equal to 1042 hours. If Hunt's utilization were to decrease from 718 hours, the SCRR model would not have to be rerun since it can be seen from the throughput change value (Column 7) that student throughput will decrease by 0.048 for each hour of decreased instructor utilization. However, since in reality, instructor resources are not applied or reduced on an hour by hour basis, the time consequence of the above statements will be examined. A review of Figure IV-2 indicates that Hunt instructs Course 461V. Forty-two (42) contact hours are required for each 461V convening. Therefore, the instructor resource will be used in 42 hour increments. A 42 hour decrease in Hunt's utilization will decrease total student

throughput by 2.0 (0.048 per hour X 42 hours). This is a logical result. Removing one 461V convening from the schedule will reduce total student throughput by 2 students, since the average class size for 461V is 2 students (Figure IV-2, Column 3).

However, a 42 hour increase in Hunt's utilization will decrease total student throughput by 4.1 students. The addition of one 461V convening would be expected to increase student throughput by 2 students, which is course 461V average class size. To determine why an additional convening will decrease throughput, first examine the limiting variable column of Figure IV-3. Instructor Ward is indicated opposite the -0.098 unit throughput change figure. The assignment listing (Figure IV-2) shows that Ward instructs courses 510G and 536P. How then is Ward related to Hunt? Examination of the Facility utilization listing (Figure IV-3) shows that both 461V and 536P utilize Classroom 180 in Building N-30, which is available 2080 hours per year. The LP resource data output in Figure IV-4 shows that N-30180 is 100 percent utilized. Therefore, the only way to increase 461V convenings is to decrease 536P convenings. The net effect of increasing 461V convenings and decreasing 536P convenings is a reduction in total student throughput.

The minimum instructor requirement for Course 461V is 504 hours, (42 hours per convening X 12 convenings per year). The difference between the Hunt's optimal utilization level of 718 hours and the minimum 461V requirements is used to increase the number of 461V convenings since Hunt is not qualified to instruct any other courses (Figure IV-2, Column 2). This increases student throughput. The ability to increase 461V convenings is meaningful only if corresponding increases in demand for course 461V can be projected. If no increased demand can be projected for 461V, then 461V convenings should be maintained at the current level. If the decision is made to maintain the current level of 461V convenings, then Hunt will be underutilized 538 hours (1042 hours availability less the 504 hours required for 461V). These 538 hours could then be devoted to cross-training Hunt to qualify to instruct additional courses (the SCRR model can assist in the identification of courses which require additional instructors) or to performing other duties.

Limiting Variable. This column (Figure IV-4, Column 8) identifies the variable whose status would change as a result of either increasing or decreasing the indicated variable. Resource variables have two status levels: (1) when utilization is equal to availability, the resource is 100 percent utilized; (2) when utilization is less than availability, the resource is not 100 percent utilized. Referring to instructor resource Hunt, the Limiting Variable column indicates that as Hunt's utilization level is decreased from 718 hours to 504 hours, the utilization of Classroom 180 in Building N-30 (N-30180) will change from status 1 (100 percent utilized) to status 2 (not 100 percent utilized). As the instructor resource level is decreased, the number of course convenings must also decrease since these two variables are directly proportional. As convenings decrease, facility utilization must also decrease, if all other variables are held constant.

Increasing instructor Hunt's utilization above the 718 hour level will cause instructor Ward to change from status level 1 to status level 2. Increasing course 461V convenings requires a decrease in course 536P convenings since both courses share Classroom 180 which is fully utilized at the indicated optimal instructor utilization rate. From Figure IV-2, Ward teaches both course 510G and 536P. Decreasing 536P convenings will, of necessity, lower Ward's annual utilization.

LP OPTIMAL SOLUTION AND SENSITIVITY ANALYSIS - COURSE DATA. The optimal number of annual course convenings which will maximize student throughput is calculated by a linear programming technique. The sensitivity analysis provides information regarding the sensitivity of the optimal solution to changes in the input data, and the solutions that result from such changes. The levels to which the class capacity and the number of annual convenings can increase and decrease before the optimal solution changes are contained in the course data section of the LP optimal solution and sensitivity analysis output. It also gives the incremental change in student throughput per unit increase or decrease in course convenings up to these levels. As with the resourse data output, the course data sensitivity analysis assumes that variables are changed individually and that the rest of the problem remains unaltered. A sample of the LP Optimal Solution and Sensitivity Analysis - Course Data output is shown in Figure IV-5.

Course CDP Number. All courses are identified by a four position alphanumeric field (Figure IV-5, Column 1).

Maximum Annual Convenings. This column (Column 2) lists the optimal number of annual course convenings for each course. The optimal number of convenings is that number of convenings which maximizes total student throughput subject to the resource limitations identified in the previous output section. Any other combination of numbers of convenings for the same set of courses is either not feasible or will result in a lower student throughput. As an example, the optimal number of convenings for course 461V is shown to be 17.1 (from Figure IV-5). Since the number of convenings is established for a one year period, fractional convenings can be interpreted to mean that the course was convened but not completed in the same year. Therefore, 17.1 convenings are equivalent to 18 convenings with the last class convened being ten percent complete at the end of the year.

Current Scheduled Convenings. The current number of scheduled convenings listed in this column (Column 3) is identical to the number of convenings stored for each course in the DOTS data base. This number is established as a lower bound for the maximum annual convening parameter discussed above. In effect, the lower bound represents an additional constraint to the LP problem. The constraint to the variable representing the number of convenings for course 461V states that 461V convenings must be greater than or equal to 12.

Average Class Size. Average class size (Column 4) is calculated by multiplying class capacity by historical course utilization. Data for class capacity and course utilization are stored in the DOTS data base. Class capacities are based upon training considerations such as student/instructor ratio; availability of training equipment, workshop, laboratory, and mock-up facilities; as well as physical considerations such as class size. Course utilization is the ratio of actual student enrollments to class capacity.

The average class size figure serves two functions. (1) Average class sizes are the LP objective function coefficients. Student throughput is established by multiplying the average class size of each course by the optimal number of convenings calculated for that course. (2) All instructor requirements are calculated based on average class size. For example, assume that a course has a 12:1 student/instructor ratio for a 10 hour segment of the course. The class capacity is 144 students. If the course utilization were 100 percent, 12 instructors would be required for that 10 hour course segment for a total of

COURSE CDP NO.	MAXIMUM ANNUAL CONVEN	CURRENT SCHEDULED CONVEN	AVERAGE CLASS SIZE	ANNUAL CONVEN RANGE MINIMUM MAXIMUM	THROUGHPUT CHANGE PER COURSE CONVEN CHANGE	CLASS SIZE RANGE MINIMUM MAXIMUM	LIMITING VARIABLE
461V	17.1	12	2	0.0	-2.0	0.0	N-30180
				24.8	-4.1	6.1	WARD
510G	18.0	6	7	12.7	-3.1	3.9	BROOKS
				26.1	-13.8	20.8	510V
510V	12.0	12	7	8.4	20.7	0.0	461V
				17.4	-20.7	27.7	HUNT
510W	57.9	24	3	0.0	-3.0	0.0	HARRISON
				57.9	NO CHG	+INF	
5107	52.0	24	8	0.0	-8.0	0.0	N-19A120
				52.0	NO CHG	+INF	
536K	73.2	50	10	55.8	-6.4	3.6	510V
				73.2	NO CHG	+INF	
536P	28.4	12	7	21.6	-4.7	2.3	WARD
				32.8	-3.7	10.7	BROOKS
5698	7.7	6	14	0.0	-14.0	0.0	OBERLE
				7.7	NO CHG	+INF	

LP OPTIMAL SOLUTION AND SENSITIVITY ANALYSIS - COURSE DATA

FIGURE IV-5

120 contact hours. However, if the course utilization were 50 percent, the instructor requirement would drop to six for the same portion of the course for a total of only 60 contact hours.

Annual Convenings Range. The annual convenings range (Column 5) indicates the upper and lower limits for the number of annual convenings parameter. The current LP solution remains optimal for all values of the convenings parameter within this range. However, as the number of annual convenings is varied from the indicated optimal, the maximum student throughput will either increase or decrease by the factor printed in the next column, throughout change per course convening change. The specified lower bound, which is the number of current scheduled convenings, is ignored in calculating the range of annual course convenings. For example, the range of annual convenings for course 536K is 55.8-73.2. The solution which maximizes student throughput is 73.2 convenings. The current number of scheduled convenings is 50 per year. If 10 convenings less than the optimal 73.2 were scheduled, the total student throughput would decrease by 64 (Column 6). Note that the student throughput drops only by 6.4 per convening, in spite of the fact that 536K class size is 10. From Figure IV-2, it is seen that course 536K and 510V utilize the same instructor. Therefore, as 536K convenings are decreased, additional convenings of 510V can be scheduled. But since 510V uses more instructor resource per convening and has a smaller class size, less convenings, each with a smaller class size, can be scheduled with the resources made available by reducing 536K convenings.

The annual convenings range (Figure IV-5, Column 5) also provides an indication of the effect of class size on the LP optimal solution. The current solution will remain optimal for class sizes in the range noted by the class size range (Column 7). As the class size drops below the capacity range minimum, the optimal number of convenings will decrease to the convening range minimum, if that quantity is greater than the current number of scheduled convenings. If the convening range minimum is less than the current scheduled convenings, then the optimal number of course convenings will be set equal to the current scheduled convenings. For example, the optimal number of convenings for course 536P is 28.4 based on a class size of 7 students. Should the class size drop below 2.3 (class size range minimum), the optimal number of convenings would decrease to 21.6 (the annual convening range minimum).

The optimal number of annual convenings for course 510G is 18.0 for a class size of 7. If the class size drops to 3.9 or less, the optimal convenings for that course will decrease to the annual convening range minimum (12.7), rather than the current scheduled convenings (6.0). The student throughput for the new optimal solution will decrease by 16.4 (decrease of 5.3 convenings X 3.1 student throughput decrease per convening).

Throughput Change Per Course Convening Change. This column (Column 6) indicates the change in the optimal student throughput which will result from a change in the number of course convenings. As long as the number of course convenings remains within the range indicated by the annual convening range column, the SCRR model need not be rerun. The effect of a change in the number of convenings can be determined by a simple calculation using the throughput change per course convening change. Again, referring to Figure IV-5 (Column 6), if course 510V convenings were to increase from the 12 per year optimal to 13 per year, the total student throughput would decrease by 20.7. From Figure IV-2, courses 510V and 536K share the same instructor (Brown). Increasing 510 convenings from

12.0 to 13.0 shifts 30 hours of Brown's time from 536K to 510V. This 30 hour shift increses 510V convenings by 1.6, but reduces possible 536K convenings by 3.2 (30 ÷ 9.3 hours per 536K convening). Therefore, student throughput will drop even though 510V convenings are increased.

Class Size Range. This column (Column 7) indicates the range of class sizes for which the indicated optimal number of course convenings will remain unchanged. A decrease in class size to below the class size minimum will cause the optimal number of convenings to decrease to either the number of current scheduled convenings or the convening range minimum, whichever is greater. An increase in class size to beyond the class size range maximum will increase the number of course convenings to the convening range maximum. Using course 461V from Figure IV-5 as an example, if the class size were increased from the current level of 2 to 6.1 or more, the optimal number of convenings would increase to 24.8 per year from the current level of 17.1.

Limiting Variable. This column (Column 8) identifies the variable which limits the convenings range and whose value will change as a result of a variation from the optimal convening level within the course convening range. Several examples from Figure IV-5 will clarify the significance of this column. As the number of course 461V convenings are decreased from the indicated optimum, N-30180 utilization will decrease from the 100 percent shown in the SCRR output. This relationship is obvious since a decline in convenings will result in lower resource requirements.

As course 536 convenings decrease from the optimal, course 510V convenings will increase from the Current Scheduled Convenings level. Instructor Brown is shared by courses 510V and 536K. A decrease in 536K convenings will increase the amount of resources available for course 510V. Decreasing 536K convenings will reduce total student throughput by 6.4 per convening deleted.

SCRR MODEL OPERATIONAL PROCEDURES

This section provides detailed procedures for executing the SCRR model, generating instructor assignment listings and facility utilization listings, modifying resource availability, and modifying course parameters which directly impact instructor requirements. Each of these procedures is initiated by keying a single word into the system via the terminal keyboard. Procedures which require the user to input data are interactive.

Figure IV-6 is an overview of the SCRR model procedures which can be invoked to perform the functions noted above. As indicated, all SCRR model input data are calculated based on the contents of the DOTS Master data base. No data inputs are required from the user. Once the model has been executed, the user has the option either to review the model output directly or to generate an instructor assignment listing which details each instructor's requirements by course. Upon reviewing the model output, the user may then elect to either modify the availability of one or more resources, to modify course parameters which determine instructor requirements, or to fix the number of course convenings at the current level. After the input data have been modified by any of the above means, the model is then reexecuted.

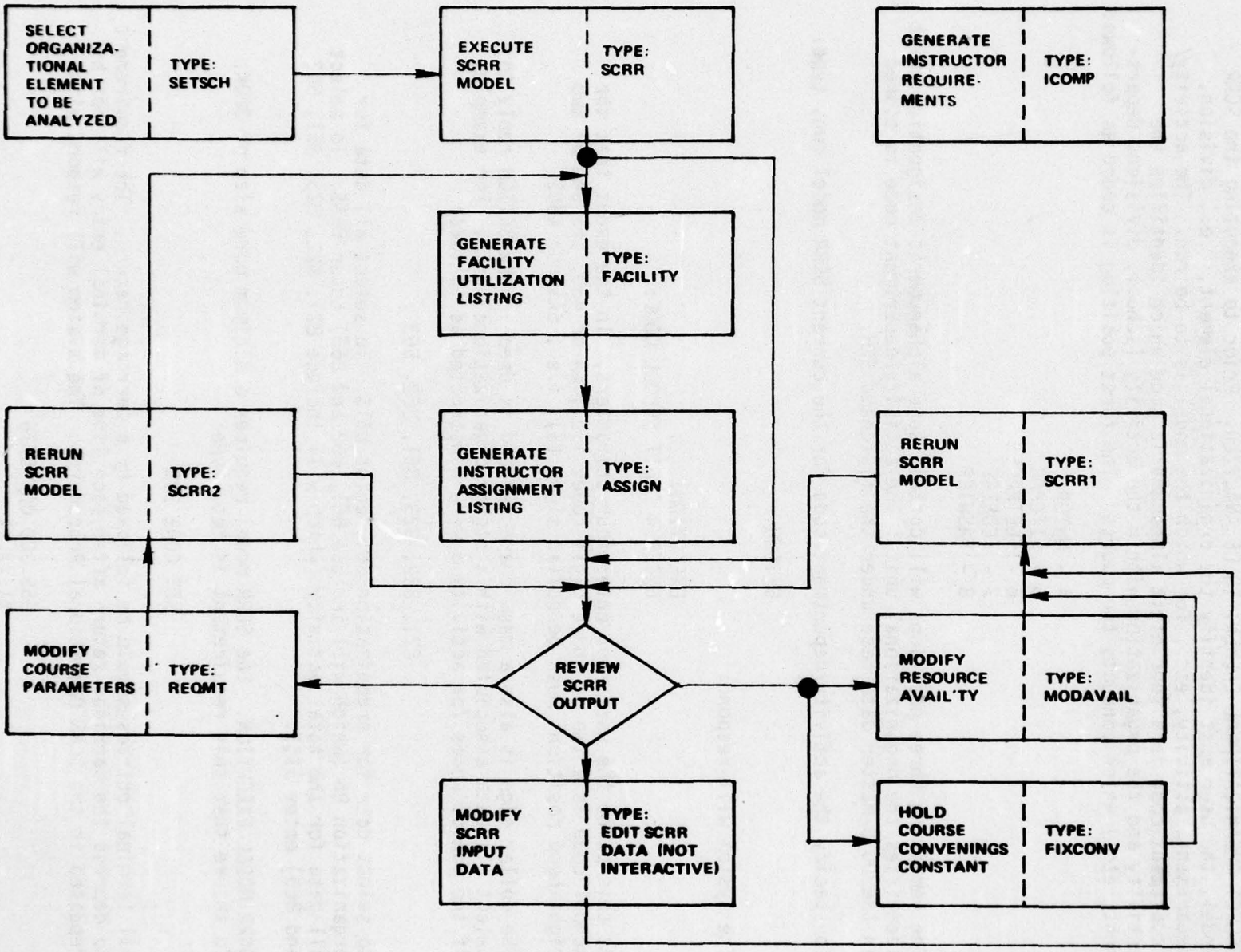


FIGURE IV-6 SCRR MODEL OPERATION OVERVIEW

SELECT ORGANIZATIONAL ELEMENT TO BE ANALYZED. Prior to executing the SCRR model, the user must identify the organizational element, i.e., division, department, activity, etc., for which the model is to be run. The activity/department code is a four digit alphanumeric code which identifies the activity and the organization within the activity (school, division, department, etc.) which conducts the course. The first position is coded as follows:

4 - NWTGP
5 - FITCPAC
6 - FLETRACEN
7 - FCDSTCP
8 - FASWTCP

The remaining three positions will contain some alphanumeric designation which identifies the organizational unit. The activity/department code is stored in the DOTS Master data base under the fieldname SCH.

To specify the activity/department code for the current SCRR model run, type:

SETSCH

The system will respond:

EXECUTION:
ENTER 4 DIGIT SCHOOL CODE:

At this point the user must enter four characters. In the event that the school code is a two or three digit code, i.e., 6A or 430, the one or two right-hand positions must be dollar signs ($), i.e., 6A$$ or 430$.

The dollar sign is also a mask character and is used in the SETSCH reply to select all data associated with a higher organizational unit. For example, if the school codes for activity 8 were constructed as follows:

821, 822, 823, 861, 862, 863

to select data for organization 861, enter 861$. To select all data for organization 86 (which will include 861, 862 and 863) enter 86$$. To select all data for the total activity (which will include 821, 822, 823, 861, 862 and 863) enter 8$$$.

SCRR MODEL EXECUTION. The SCRR model requires a minimum core size of 320K. To ensure that this requirement is met, type:

SET CORE 320

All terminal entries should be followed by a carriage return. The requirement to depress the carriage return after each line of terminal entry will not be repeated in the SCRR Operational Procedures. The system will respond with:

CSS.300 06/15/76
XX.XX.XX

where XX.XX.XXis the time-of-day in hours, minutes and seconds.

To execute the SCRR model, type:

SCRR

The SCRR executive will retrieve the required data elements from the three DOTS Master data base files, make additional computations, format resulting data for linear programming (LP) optimization, perform the LP optimization, and format the LP results for output. The system will respond with the following message:

```
THIS PROGRAM PERFORMS THE LINEAR PROGRAMMING OPTIMIZATION
OF THE RESOURCE DATA CURRENTLY STORED IN THE MASTER DATA FILES
FOR THE DEPARTMENT IDENTIFIED THROUGH EXECUTION OF THE SETSCH
COMMAND.  NO FURTHER USER INTERVENTION WILL BE REQUIRED.
DO YOU WISH TO PROCEED?  (Y/N)
```

At this point, the user should verify, by checking prior terminal entries, that the SETSCH command has been invoked and the desired activity/department code has been entered. If nothing has been overlooked and the user wishes to proceed, type:

Y

If the user wishes to terminate model execution at this point, type:

N

The system will respond with the time-of-day message.

If the proceed option is answered affirmatively, the system will respond after a short pause with the following message:

```
MPS III, ON-LINE VERSION, READY.
MPS COMMAND:
MPS TERMINATING.
TO PRINT SCRR MODEL OUTPUT, TYPE:
P SCRR OUTPUT
```

The MPS messages indicate first that the Mathematical Programming System software package has been attached to the operating system, that the LP optimization calculations are being performed, and finally that the LP segment of the model is complete.

The SCRR executive does not automatically print the model output. As the message indicates, the output is printed by typing:

P SCRR OUTPUT

The SCRR output is contained on a permanent file. Printing the output does not erase the file. The file SCRR OUTPUT will exist and can be printed until the SCRR model is reexecuted. Reexecution of the model will cause the new output to be written on the file SCRR OUTPUT.

A sample SCRR output file follows:

RESOURCE	ANNUAL NON-CLASSROOM HOURS	ANNUAL AVAIL CLASSROOM CONTACT HOURS	ANNUAL REQMT CLASSROOM CONTACT HOURS	REQMT VS AVAIL RATIO	RESOURCE UTIL. RANGE MINIMUM MAXIMUM	THROUGHPUT CHANGE PER UNIT RESOURCES CHANGE	LIMITING VARIABLE
DORSEY	838	1042	681	65	519.0 681.0	-0.688 NO CHG	51@V
HUNT	838	1042	718	68	504.0 1504.0	-0.048 -0.098	N-3@18@ PAUL
STEWART	838	1042	1040	99	480.0 1040.0	-0.400 NO CHG	N19A12@
WARD	838	1042	1042	100	880.0 1042.0	-0.196 NO CHG	PAUL
BROWN	839	1041	1041	100	879.0 1148.0	-0.922 0.922	HUNT 461V
PAUL	838	1042	1042	100	880.0 1042.0	-0.196 0.196	HUNT WARD
STREIT	838	1042	1040	99	480.0 1040.0	-0.400 NO CHG	N19A12@
ROHLFING	838	1042	1040	99	480.0 1040.0	-0.400 NO CHG	N19A12@
BROOKS	838	1042	1042	100	935.0 1204.0	-0.154 0.154	461V HUNT
SPARKES	838	1042	1040	99	480.0 1040.0	-0.400 NO CHG	N19A12@
HARRISON	838	1042	1042			-0.167	

COURSE CDP NO.	MAXIMUM ANNUAL CONVEN	CURRENT SCHEDULED CONVEN	AVERAGE CLASS SIZE	ANNUAL CONVEN RANGE MINIMUM	ANNUAL CONVEN RANGE MAXIMUM	THROUGHPUT CHANGE PER COURSE CONVEN CHANGE	CLASS SIZE RANGE MINIMUM	CLASS SIZE RANGE MAXIMUM	LIMITING VARIABLE
461V	17.1	12	2	Ø.Ø	24.8	-2.Ø -4.1	Ø.Ø	6.1	N-3Ø18Ø WARD
51ØG	18.Ø	6	7	12.7	26.1	-3.1 -13.8	3.9	2Ø.8	BROOKS 51ØV
51ØV	12.Ø	12	7	8.4	17.4	2Ø.7 -2Ø.7	Ø.Ø	27.7	461V HUNT
51ØW	57.9	24	3	Ø.Ø	57.9	-3.Ø NO CHG	Ø.Ø	+INF	HARRISON
51Ø7	52.Ø	24	8	Ø.Ø	52.Ø	-8.Ø NO CHG	Ø.Ø	+INF	N-19A12Ø
536K	73.2	5Ø	1Ø	55.8	73.2	-6.4 NO CHG	3.6	+INF	51ØV
536P	28.4	12	7	21.6	32.8	-4.7 -3.7	2.3	1Ø.7	WARD BROOKS
5698	7.7	6	14	Ø.Ø	7.7	-14.Ø NO CHG	Ø.Ø	+INF	OBERLE

INSTRUCTOR ASSIGNMENT LISTING GENERATION. To generate a detailed instructor assignment listing for all instructors appearing in the SCRR OUTPUT, type:

ASSIGN

The system will respond:

EXECUTION:
TO PRINT INSTRUCTOR ASSIGNMENTS, TYPE:
P ASSIGN LISTING

To print the assignment listing, type:

P ASSIGN LISTING

As with the SCRR OUTPUT, the ASSIGN LISTING is a permanent file and can be printed as often as is required until the ASSIGN executive is repeated, at which time the old listing file will be replaced by the new listing file.

NOTE: The ASSIGN executive should not be invoked until either the SCRR model has been executed or the executive RAMSCRR has been executed.

To execute RAMSCRR, type:

RAMSCRR

after executing SETSCH. RAMSCRR extracts the data required to execute ASSIGN from the DOTS Master data base. To execute the SCRR model after executing RAMSCRR, type:

SCRR3

A sample instructor assignment listing is shown below.

INSTRUCTOR	COURSE	CLASIZE	CONV	HRS/CONV	TOT HRS	AVAIL HRS
BROOKS	51ØG	7	6	2Ø.Ø	12Ø.Ø	
	53ØK	1Ø	5Ø	9.3	465.Ø	
				TOT =	585.Ø	1Ø42
BROWN	51ØV	7	12	3Ø.Ø	36Ø.Ø	
	536K	1Ø	5Ø	9.3	465.Ø	
				TOT =	825.Ø	1Ø41
DORSEY	536K	1Ø	5Ø	9.3	465.Ø	1Ø42
HARRISON	51ØW	3	24	18.Ø	432.Ø	1Ø42
HUNT						

FACILITY UTILIZATION LISTING GENERATION. To generate a detailed facility utilization listing for all facilities appearing in the SCRR OUTPUT, type:

FACILITY

The system will respond:

EXECUTION:
TO PRINT FACILITY LISTING, TYPE:
P FACILITY LISTING

To print the facility listing, type:

P FACILITY LISTING

As with the SCRR OUTPUT and ASSIGN LISTING, the FACILITY LISTING is a permanent file and can be printed as often as is required until the FACILITY executive is repeated. When the FACILITY executive is executed the previous listing file stored will be replaced by the current listing file.

NOTE: The FACILITY executive should not be invoked until either the SCRR model has been executed or the executive RAMSCRR has been executed. To execute RAMSCRR, type:

RAMSCRR

after executing SETSCH. RAMSCRR extracts the data required to execute FACILITY from the DOTS Master data base. To execute the SCRR model after executing RAMSCRR, type:

SCRR3

A sample facility utilization listing is shown below:

BLDG / RM	COURSE	CLASIZE	CONV	HRS/CONV	TOT HRS	AVAIL HRS
FLTLHGR	51ØW	3	24	12.Ø	288.Ø	2Ø8Ø
N-3Ø1Ø6	51ØG	7	6	33.Ø	198.Ø	2Ø8Ø
N-3Ø1Ø7	51ØG	7	6	27.Ø	162.Ø	
	5698	14	6	11Ø.Ø	66Ø.Ø	
				TOT =	822.Ø	2Ø8Ø
N-3Ø1Ø8	5698	14	6	4Ø.Ø	24Ø.Ø	2Ø8Ø
N-3Ø167	536K	1Ø	5Ø	2Ø.Ø	1ØØØ.Ø	2Ø8Ø
N-3Ø18Ø	461V	2	12	42.Ø	5Ø4.Ø	
	536P	7	12	48.Ø	576.Ø	
				TOT =	1Ø8Ø.Ø	2Ø8Ø
N-3Ø181	51ØV	7	12	3Ø.Ø	36Ø.Ø	
	51ØW	3	24	6.Ø	144.Ø	
				TOT =	5Ø4.Ø	2Ø8Ø

24

RESOURCE AVAILABILITY MODIFICATION. To modify the availability of an instructor or a facility, type:

MODAVAIL

The system will respond:

EXECUTION:
ENTER NAME OF INSTRUCTOR OR FACILITY:

At this point the user should enter the name of an instructor or a facility (building/room) exactly as it appears in the SCRR OUTPUT. The system will respond:

ENTER CURRENT AVAILABLE HOURS, NEW AVAILABLE HOURS:

The user should now enter the number of available hours shown in the SCRR OUTPUT for the instructor or facility named above, followed by either a comma or a space, and the new availability figure. The number of available hours must be less than 9,999. The system will then respond either:

AVAIL ID 1 XXXX ID 2 XXXX

or

AVAIL ID 1 XXXX

(where either ID 1 or ID 2 is the ID number associated with the name entered above and XXXX is the current availability) followed by either:

AVAIL ID 1 XXXX ID 2 AAAA

or

AVAIL ID 1 AAA

(where AAAA) is the new availability).

If the name is not entered correctly, the system will respond:

EOF:
NO CHANGE

If the current available hour figure is not entered correctly, the system will respond either:

AVAIL ID 1 XXXX ID 2 XXXX

or

AVAIL ID 1 XXXX

followed by:

NO CHANGE

After the status of the first availability change request has been printed, the system will respond:

ANY MORE CHANGES? (YES/NO)

If the respond is YES, the program will recycle and again request the name of the resource to be changed. If the response is NO, the system will respond:

CHANGES TO AVAILABLE HOURS HAVE BEEN INCORPORATED.
TO RERUN SCRR MODEL, TYPE: SCRR1

NOTE: The MODAVAIL executive should not be invoked until after the SCRR model has been executed.

CONSTANT COURSE CONVENINGS/OPTIMAL COURSE CONVENINGS MODIFICATION. The SCRR model can be executed under two conditions: (1) maximize the number of course convenings based on the level of resources available; or (2) specify the number

of course convenings to be constant at the current number of convenings scheduled. Executing the SCRR model under condition (2) will provide the user with an accurate summary of current resource utilization.

To execute the SCRR model under condition (1) as explained above, type:

MAXCONV

The system will respond:

DATA MODIFIED TO OPTIMIZE COURSE CONVENINGS.
TO RERUN THE SCRR MODEL, TYPE: SCRR1

To execute the SCRR model under condition (2) as explained above, first type:

FIXCONV

The system will respond:

COURSE CONVENINGS FIXED AT THE CURRENT LEVEL.
TO RERUN THE SCRR MODEL, TYPE: SCRR1

As directed in the above message, to rerun the SCRR model after invoking either MAXCONV OR FIXCONV, type:

SCRR1

After the system response is printed, type:

P SCRR OUTPUT

to print the SCRR model output.

NOTE: FIXCONV or MAXCONV should not be invoked until after the SCRR model has been executed.

COURSE PARAMETER MODIFICATION. To modify course utilization, class capacity, or course length, type:

REQMT

The system will respond with the following message:

DO YOU WISH INSTRUCTIONS TO BE PRINTED? (YES/NO)

If the user responds NO, no directions will be printed. However, requests for data entry will be printed. If the user responds YES, the following message will be printed:

```
THIS PROGRAM PROVIDES THE CAPABILITY TO MODIFY COURSE
UTILIZATION, CLASS CAPACITY, AND COURSE LENGTH EXPRESSED
IN CONTACT HOURS TO ASSESS THE IMPACT ON TOTAL INSTRUCTOR
REQUIREMENTS.
*  TO INCREASE UTILIZATION 2Ø ENTER: .2Ø
*  TO DECREASE UTILIZATION 1Ø ENTER: -.1Ø
*  IF NO CHANGE IS DESIRED ENTER: Ø
ENTER UTILIZATION DELTA
```

The user should enter the utilization delta as requested. The utilization percentage for all courses taught by the organization identified in the SETSCH executive will be modified by the utilization delta entry. For example, if the current utilization of two courses is .40 and .60, and the utilization delta entered is .5, the new course utilizations will be .60 and .90.

After the course utilization delta has been entered, the system will respond:

```
*  TO INCREASE CLASS CAPACITIES 2Ø% ENTER: .2Ø
*  TO DECREASE CLASS CAPACITIES 1Ø% ENTER: -.1Ø
*  IF NO CHANGE IS DESIRED ENTER:  Ø
ENTER CLASS CAPACITY DELTA:

*  TO INCREASE COURSE LENGTH 2Ø% ENTER: .2Ø
*  TO DECREASE COURSE LENGTH 1Ø% ENTER: -.1Ø
*  IF NO CHANGE IS DESIRED ENTER: Ø
ENTER COURSE LENGTH DELTA:
```

The class capacity and course length deltas should be entered as requested. After the last data entry the system will respond:

```
REQUESTED CHANGES HAVE BEEN INCORPORATED.
TO RERUN SCRR MODEL, TYPE:  SCRR2
```

As the above message indicates, to rerun the SCRR model after executing REQMT, type:

```
SCRR2
```

NOTE: The REQMT executive should not be invoked until after the SCRR model has been executed.

ADDITIONAL DATA MODIFICATION. To modify SCRR input data not specifically addressed in either the MODAVAIL executive or the REQMT executive requires the use of the VP/CSS edit command. The edit command allows the user to add, delete or change data stored in any data file. The user should refer to NCSS form number 108-4 effective for release CSS.300 for detailed directions on use of the edit command. The SCRR model input data is stored in the file SCRR DATA.

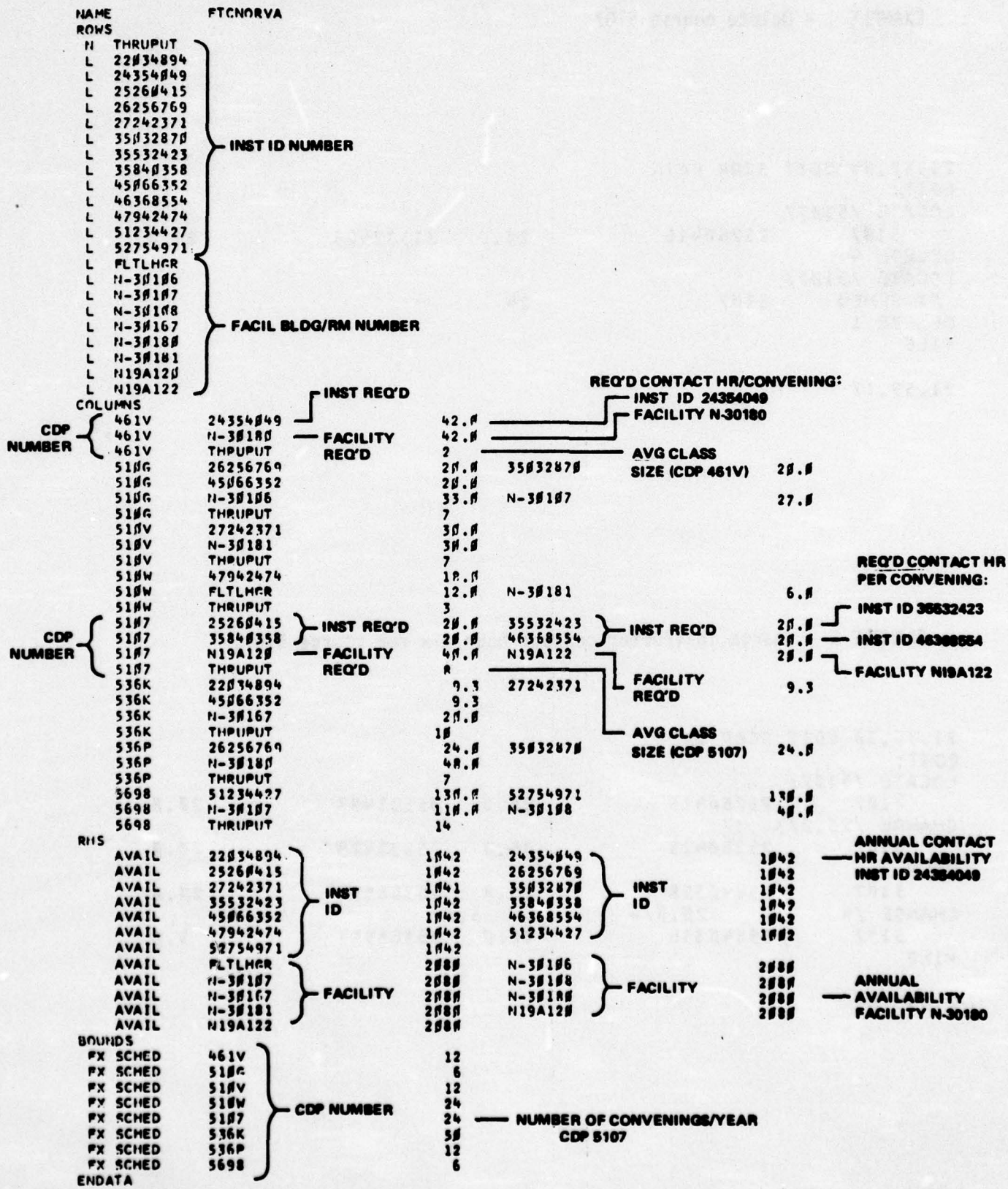
To modify any data contained in SCRR DATA, type:

```
EDIT SCRR DATA
```

The system will respond:

```
EDIT:
```

An annotated listing of a SCRR DATA file is shown below:



Two examples of possible changes to SCRR DATA are presented below.

EXAMPLE 1 - Delete course 5107

```
21.52.Ø4 EDIT SCRR DATA
EDIT:
LOCATE /51Ø7/
    51Ø7        2526Ø415          2Ø.Ø    35532423          2Ø.Ø
DELETE 4
LOCATE /51Ø7/
 FX SCHED       51Ø7              24
DELETE 1
FILE

21.55.17
```

EXAMPLE 2 - Change instructor contact hour mix for course 5107.

```
21.55.3Ø EDIT SCRR DATA
EDIT:
LOCATE /51Ø7/
    51Ø7        2526Ø415          2Ø.Ø    35532423          2Ø.Ø
CHANGE /2Ø.Ø/35.Ø/
    51Ø7        2526Ø415          35.Ø    35532423          2Ø.Ø
DOWN
    51Ø7        3584Ø358          2Ø.Ø    46368554          2Ø.Ø
CHANGE /4           2Ø.Ø/4              5.Ø/
    51Ø7        3584Ø358          2Ø.Ø    46368554           5.Ø
FILE

21.59.54
```

INSTRUCTOR REQUIREMENTS COMPUTATION

To generate instructor requirements based on the computational technique outlined in CNTECHTRAINST 5311.1A CH-1 and the data currently stored in the DOTS data files, type:

ICOMP

The system will respond:

TO CALCULATE INSTRUCTOR REQUIREMENTS FOR:
***A SINGLE COURSE TYPE: CDP
***ALL COURSES WITHIN A DIVISION OR DEPT TYPE: SCH

if CDP is entered the system will respond:

ENTER COURSE CDP NUMBER:

Upon entry of the CDP number the system will respond:

TO CHANGE THE ANNUAL DEMAND FOR COURSE XXXX FROM THE CURRENT LEVEL AS STORED IN THE DATA BASE TO A NEW VALUE, TYPE NEW DEMAND VALUE; ELSE TYPE Ø.

Upon entry of demand figure the system will print the computed instructor requirements for the identified course.

If SCH is entered in response to the initial program data request the system will reply:

ENTER DIVISION/DEPT CODE:

Upon entry of the DIVISION/DEPT CODE the system will print the computed instructor requirements for all courses in the identified division/dept.

A sample instructor requirement computation output is shown below:

COMPUTED INSTRUCTOR REQUIREMENTS 08/31/76

CDP	INSTR REQD CUR DMD
2252	.20
2253	.62
2275	.09
2304	.13
9338	.43
9339	.11
9341	.72
9342	.35
9343	.44
9344	.76
9345	.55
9459	.09
9774	.27

SCRR MODEL EXECUTION PROCEDURES SUMMARY

MODEL FUNCTION	KEYBOARD ENTRY
Select organization	SETSCH
Execute SCRR model	SCRR
Extract data from data base to run ASSIGN or FACILITY	RAMSCRR
Run SCRR model after RAMSCRR	SCRR3
Generate instructor assignment listing (SCRR or RAMSCRR must be executed first)	ASSIGN
Generate facility assignment listing (SCRR or RAMSCRR must be executed first)	FACILITY
Modify resource (instructor or facility) availability	MODAVAIL
Fix the number of course convenings at the current level (do not optimize)	FIXCONV
Modify SCRR DATA to optimize course convenings	MAXCONV
Rerun SCRR model after MODAVAIL, FIXCONV, or MAXCONV	SCRR1
Modify course parameters (course utilization, class capacity, course length)	REQMT
Rerun SCRR model after REQMT	SCRR2
Modify additional model input data	EDIT SCRR DATA
Instructor Requirements Computation	ICOMP

SCRR MODEL OPERATIONAL TEST PROCEDURES

The SCRR operational test has been designed to exercise all SCRR model software. Since, operationally, all SCRR model input is computed based on the contents of the DOTS Data Base, a miniature version of the DOTS data base has been created to test the model/data base interface software. This Test data base will provide a constant data input to the SCRR model. The operational test will be considered successful if the model output which is generated as a result of following the procedures outlined below, is identical to the model output printed in this section. In addition to the operational test procedures and the operational test output, a description of the Test data base is also included below.

SCRR TEST DATA BASE DESCRIPTION: The Test data base is identical to the DOTS data base in every way but size. Like the DOTS data base, the Test base is divided into three files: the course file (TESTC); the instructor file (TESTI); and the facility file (TESTF). To minimize storage costs and execution time, most fields not required for SCRR model input have been omitted from the Test data base. The descriptions of the three Test data base files are presented below:

RAMASTER DESCRIPTION FOR
RAMIS FILE NAMED TESTC
AT 9.14.45 ON 08/27/76

LIST	FIELD NAME	SYNONYM	LEV	TYPE	LEN	SEG	SFLAG
1	CDP	CDP	1	A	4	10	S
2	SCH	SCH	1	A	4	10	S
3	DEL	DEL	1	A	1	10	S
4	CONV1	CONV1	1	I	3	10	S
5	CAP1	CAP1	1	I	3	10	S
6	UCAP	UCAP	1	I	3	10	S
7	F1	F1	2	I	9	5	S
8	F2	F2	2	I	4	5	S
9	F3	F3	2	I	1	5	S
10	NME	NME	3	A	16	1	V

RAMASTER DESCRIPTION FOR
RAMIS FILE NAMED TESTF
AT 9.19.07 ON 08/27/76

LIST	FIELD NAME	SYNONYM	LEV	TYPE	LEN	SEG	SFLAG
1	CDP	CDP	1	A	4	10	S
2	SCH	SCH	1	A	4	10	S
3	BLDG	BLDG	2	A	4	2	S
4	RM	RM	2	A	4	2	S
5	RMCAP	RMCAP	2	I	3	2	S
6	RMTP	RMTP	2	A	1	2	S
7	REQHRS	REQHRS	2	F	6.1	2	S
8	AVHRS	AVHRS	2	I	5	2	S

RAMASTER DESCRIPTION FOR
RAMIS FILE NAMED TEST1
AT 9.16.26 ON 08/27/76

LIST	FIELD NAME	SYNONYM	LEV	TYPE	LEN	SEG	SFLAG
1	ID	ID	1	I	9	15	S
2	NME	NME	1	A	16	15	S
3	SCH	SCH	1	A	4	15	S
4	ROT	ROT	1	I	4	15	S
5	CONT	CONT	1	I	4	15	S
6	CDEV	CDEV	1	I	4	15	S
7	EQMT	EQMT	1	I	4	15	S
8	PREP	PREP	1	I	4	15	S
9	BKIN	BKIN	1	I	4	15	S
10	CTRN	CTRN	1	I	4	15	S
11	FTRN	FTRN	1	I	4	15	S
12	MOUT	MOUT	1	I	4	15	S
13	ISCH	ISCH	1	I	4	15	S
14	FLET	FLET	1	I	4	15	S
15	ADMN	ADMN	1	I	4	15	S
16	ANLV	ANLV	1	I	4	15	S
17	SKLV	SKLV	1	I	4	15	S
18	SLIB	SLIB	1	I	4	15	S

To complete the description of SCRR model data input for the operational test, the contents of the three test data files are included on the following pages. The contents of the test data files will remain constant.

FACILITY FILE (TESTF)

CDP	BLDG	RM	RMCAP	RMTP	REQHRS	AVHRS
461V	N-30	180	10	3	42.0	2080
510G	N-30	106	12	1	33.0	2080
		107	18	2	27.0	2080
510V	N-30	181	10	3	30.0	2080
510W	FLTL	HGR	10	2	12.0	2080
	N-30	181	10	3	6.0	2080
5107	N19A	120	16	1	40.0	2080
		122	16	2	20.0	2080
536K	N-30	167	10	3	20.0	2080
536P	N-30	180	10	3	48.0	2080
5698	N-30	107	18	2	110.0	2080
		108	18	1	40.0	2080

TAEG REPORT NO. 36

COURSE FILE (TESTC)

CDP	CONV1	CAP1	UCAP	NME	ID-SIR	QUAL-CHRS	ASN-TYP
461V	12	1Ø	2Ø		5	18	2
					1Ø	24	1
				HUNT	243548492	1ØØ	1
51ØG	6	12	6Ø		6	27	2
					12	33	1
				WARD	262567698	1ØØ	1
				PAUL	358328788	1ØØ	1
				BROOKS	458663526	1ØØ	1
51ØV	12	1Ø	65		1Ø	3Ø	3
				BROWN	272423719	1ØØ	1
51ØW	24	1Ø	25		1Ø	18	3
				HARRISON	479424748	1ØØ	1
51Ø7	24	16	5Ø		5	2Ø	2
					16	4Ø	1
				STEWART	2526Ø4158	1ØØ	1
				STREIT	355324236	1ØØ	1
				ROHLFING	3584Ø3587	1ØØ	1
				SPARKES	463685542	1ØØ	1
536K	5Ø	1Ø	95		5	8	2
					1Ø	12	1
				DORSEY	22Ø348942	1ØØ	1
				BROWN	272423719	1ØØ	1
				BROOKS	458663526	1ØØ	1
536P	12	8	9Ø		8	48	3
				WARD	262567698	1ØØ	1
				BROWN	272423719	1ØØ	Ø
				PAUL	358328788	1ØØ	1
5698	6	18	8Ø		6	11Ø	2
					18	4Ø	1
				OBERLE	512344273	1ØØ	1
				WAGNER	527549719	1ØØ	1

INSTRUCTOR FILE (TESTI)

ID	NME	ROT	CONT	CDEV	EOMT	PREP	BKIN	CTRN	FTRN	MDUT
220348942	DORSEY	7801	1042	0	500	0	100	0	238	0
243540492	HUNT	7610	1042	0	500	0	100	0	238	0
252604158	STEWART	7907	1042	0	500	0	100	0	238	0
262567698	WARD	7711	1042	0	500	0	100	0	238	0
272423719	BROWN	7708	1041	0	500	0	100	0	239	0
350328700	PAUL	7704	1042	0	500	0	100	0	238	0
355324236	STREIT	7808	1042	0	500	0	100	0	238	0
358403587	ROHLFING	7811	1042	0	500	0	100	0	239	0
450663526	BROOKS	7611	1042	0	500	0	100	0	238	0
463685542	SPARKES	7711	1042	0	500	0	100	0	238	0
479424748	HARRISON	7902	1042	0	500	0	100	0	238	0
512344273	OBERLE	7612	1002	0	500	0	100	0	278	0
527549719	WAGNER	7908	1042	0	500	0	100	0	238	0

OPERATIONAL TEST PROCEDURES. This section presents the detailed steps required to perform the SCRR model operational test. These procedures assume that the user has successfully completed the NCSS login procedures.

a) To initialize the operational test, type:

SCRRTEST

followed by a carriage return. The SCRRTEST routine modifies the SCRR software so that model input data are computed from the Test data files TESTC, TESTI, and TESTF, rather than the DOTS data base files COURSE, INST, and FACIL.

Upon successful completion, the system will respond with the time-of-day in hours, minutes, and seconds (XX.XX.XX).

b) The SCRR model requires a minimum computer core size of 320K. To ensure that this requirement is met, type:

SET CORE 320

followed by a carriage return. The system will respond with

CSS.300 06/15/76
XX.XX.XX

c) To initialize SCRR model processing, type:

SCRR

followed by a carriage return. The SCRR executive will retrieve the required data elements from the three Test data files, make additional computations, format resulting data for linear programming (LP) optimization, perform LP optimization, and format LP results for output. The system will respond with the following message:

THIS PROGRAM PERFORMS THE LINEAR PROGRAMMING OPTIMIZATION
OF THE RESOURCE DATA CURRENTLY STORED IN THE MASTER DATA FILES
FOR THE DEPARTMENT IDENTIFIED THROUGH EXECUTION OF THE SETSCH
COMMAND. NO FURTHER USER INTERVENTION WILL BE REQUIRED.
DO YOU WISH TO PROCEED? (Y/N)

At this point, the user should verify that SCRRTEST has been entered prior to typing SCRR. If SCRRTEST has been entered, the user should respond by typing:

YES

followed by a carriage return. If SCRRTEST was not previously entered, the user should reply:

NO

followed by a carriage return. The system will respond with a time-of-day message, at which point step a) of this procedure should be repeated.

If the proceed option is answered affirmatively, the system will respond after a short pause with the following messages:

```
MPS III, ON-LINE VERSION, READY.
MPS COMMAND:
MPS III TERMINATING.
TO PRINT SCRR MODEL OUTPUT, TYPE:
P SCRR OUTPUT
```

The MPS messages indicate first that the Mathematical Programming System software package has been attached to the operating system, then that the optimization calculations are being performed, and finally that the optimization is complete.

The SCRR executive does not automatically print the model output. As the message indicates to print the SCRR model output, type:

P SCRR OUTPUT

Typing P SCRR OUTPUT will cause the following output to appear on the screen. Steps a), b), and c) of this procedure can be considered successful only if the following output is exactly duplicated.

COURSE CDP NO.	MAXIMUM ANNUAL CONVEN	CURRENT SCHEDULED CONVEN	AVERAGE CLASS SIZE	ANNUAL CONVEN RANGE MINIMUM MAXIMUM	THROUGHPUT CHANGE PER COURSE CONVEN CHANGE	CLASS SIZE RANGE MINIMUM MAXIMUM	LIMITING VARIABLE
461V	17.1	12	2	0.0	-2.0	0.0	N-30180
				24.8	-4.1	6.1	WARD
510G	18.0	6	7	12.7	-3.1	3.9	BROOKS
				26.1	-13.8	20.8	510V
510V	12.0	12	7	8.4	20.7	0.0	461V
				17.4	-20.7	27.7	HUNT
510W	57.9	24	3	0.0	-3.0	0.0	HARRISON
				57.9	NO CHG	+INF	
5107	52.0	24	8	0.0	-8.0	0.0	N-19A120
				52.0	NO CHG	+INF	
536K	73.2	50	10	55.8	-6.4	3.6	510V
				73.2	NO CHG	+INF	
536P	28.4	12	7	21.6	-4.7	2.3	WARD
				32.8	-3.7	10.7	BROOKS
5698	7.7	6	14	0.0	-14.0	0.0	OBERLE
				7.7	NO CHG	+INF	

RESOURCE	ANNUAL NON-CLASSROOM HOURS	ANNUAL AVAIL CLASSROOM CONTACT HOURS	ANNUAL REQMT CLASSROOM CONTACT HOURS	REQMT VS AVAIL RATIO	RESOURCE UTIL. RANGE MINIMUM MAXIMUM	THROUGHPUT CHANGE PER UNIT RESOURCES CHANGE	LIMITING VARIABLE
DORSEY	838	1040	681	65	519.0 681.0	-0.688 NO CHG	510V
HUNT	838	1040	718	68	504.0 1504.0	-0.048 -0.098	N-301800 PAUL
STEWART	838	1040	1040	99	480.0 1040.0	-0.400 NO CHG	N19A120
WARD	838	1040	1040	100	880.0 1042.0	-0.196 NO CHG	PAUL
BROWN	839	1041	1041	100	879.0 1148.0	-0.922 0.922	HUNT 461V
PAUL	838	1040	1042	100	880.0 1042.0	-0.196 0.196	HUNT WARD
STREIT	838	1040	1040	99	480.0 1040.0	-0.400 NO CHG	N19A120
ROHLFING	838	1040	1040	99	480.0 1040.0	-0.400 NO CHG	N19A120
BROOKS	838	1040	1042	100	935.0 1204.0	-0.154 0.154	461V HUNT
SPARKES	838	1040	1040	99	480.0 1040.0	-0.400 NO CHG	N19A120
HARRISON	838	1040	1042	100	432.0 3120.0	-0.167 0.167	510W FLTLHGR
OBERLE	878	1002	1002	100	780.0 1042.0	-0.108 0.108	5698 WAGNER
WAGNER	838	1042	1002	96	780.0 1002.0	-0.108 NO CHG	OBERLE
FLTLHGR		2080	695	33	288.0 694.7	-0.250 NO CHG	HARRISON
N-301106		2080	596	28	419.1 862.9	-0.093 -0.417	BROOKS 510V
N-301107		2080	1335	64	1190.7 1553.9	-0.114 -0.510	BROOKS 510V
N-301108		2080	308	14	240.0 308.3	-0.350 NO CHG	OBERLE
N-301167		2080	1465	70	1116.1 1464.5	-0.320 NO CHG	510V
N-301180		2080	2080	100	1866.0 2404.0	-0.048 0.048	461V HUNT
N-301181		2080	707	34	504.0 869.3	-0.500 -0.688	HARRISON 510V
N19A120		2080	2080	100	960.0 2084.0	-0.200 0.200	5107 SPARKES
N19A122		2080	1040	50	480.0 1040.0	-0.400 NO CHG	N19A120

d) to generate a detailed instructor assignment listing for all instructors appearing in the SCRR OUTPUT, type:

ASSIGN

followed by a carriage return. The system will respond:

EXECUTION:
TO PRINT INSTRUCTOR ASSIGNMENTS, TYPE:
P ASSIGN LISTING

Then type:

P ASSIGN LISTING

Operational test procedure d) is successful if the following output is duplicated exactly.

INSTRUCTOR	COURSE	CLASIZE	CONV	HRS/CONV	TOT HRS	AVAIL HRS
BROOKS	51ØG	7	6	2Ø.Ø	12Ø.Ø	
	536K	1Ø	5Ø	9.3	465.Ø	
				TOT =	585.Ø	1Ø42
BROWN	51ØV	7	12	3Ø.Ø	36Ø.Ø	
	536K	1Ø	5Ø	9.3	465.Ø	
				TOT =	825.Ø	1Ø41
DORSEY	536K	1Ø	5Ø	9.3	465.Ø	1Ø42
HARRISON	51ØW	3	24	18.Ø	432.Ø	1Ø42
HUNT	461V	2	12	42.Ø	5Ø4.Ø	1Ø42
OBERLE	5698	14	6	13Ø.Ø	78Ø.Ø	1ØØ2
PAUL	51ØG	7	6	2Ø.Ø	12Ø.Ø	
	536P	7	12	24.Ø	288.Ø	
				TOT =	4Ø8.Ø	1Ø42
ROHLFING	51Ø7	8	24	2Ø.Ø	48Ø.Ø	1Ø42
SPARKES	51Ø7	8	24	2Ø.Ø	48Ø.Ø	1Ø42
STEWART	51Ø7	8	24	2Ø.Ø	48Ø.Ø	1Ø42
STREIT	51Ø7	8	24	2Ø.Ø	48Ø.Ø	1Ø42
WAGNER	5698	14	6	13Ø.Ø	78Ø.Ø	1Ø42
WARD	51ØG	7	6	2Ø.Ø	12Ø.Ø	
	536P	7	12	24.Ø	288.Ø	
				TOT =	4Ø8.Ø	1Ø42

e) To fix the number of course convenings per year at the current level, type:

FIXCONV

followed by a carriage return. Upon successful completion, the system will respond:

COURSE CONVENINGS FIXED AT THE CURRENT LEVEL.
TO RERUN SCRR MODEL, TYPE: SCRR1

As directed in the above message to return the SCRR model, type:

SCRR1

followed by a carriage return. After the system response is printed, type:

P SCRR OUTPUT

to print SCRR model output. Test procedure e) is successful if the following output is duplicated exactly.

COURSE CDP NO.	MAXIMUM ANNUAL CONVEN	CURRENT SCHEDULED CONVEN	AVERAGE CLASS SIZE	ANNUAL CONVEN RANGE MINIMUM MAXIMUM	THROUGHPUT CHANGE PER COURSE CONVEN CHANGE	CLASS SIZE RANGE MINIMUM MAXIMUM	LIMITING VARIABLE
461V	12.Ø	12	2	7.1 24.8	4.1 -4.1	Ø.Ø 6.1	WARD HUNT
51ØG	7.6	6	7	Ø.Ø 12.7	-7.Ø -9.8	Ø.Ø 16.8	BROOKS 51ØV
51ØV	12.Ø	12	7	1Ø.9 15.4	14.8 -14.8	Ø.Ø 21.8	51ØG WARD
51ØW	57.9	24	3	Ø.Ø 57.9	-3.Ø NO CHG	Ø.Ø +INF	HARRISON
51Ø7	52.Ø	24	8	Ø.Ø 52.Ø	-8.Ø NO CHG	Ø.Ø +INF	N-19A12Ø
536K	95.7	5Ø	1Ø	84.7 95.7	-4.6 NO CHG	5.4 +INF	51ØV
536P	32.8	12	7	21.6 32.8	-4.7 NO CHG	2.3 +INF	461V
5698	7.7	6	14	Ø.Ø 7.7	-14.Ø NO CHG	Ø.Ø +INF	OBERLE

RESOURCE	ANNUAL NON-CLASSROOM HOURS	ANNUAL AVAIL CLASSROOM CONTACT HOURS	ANNUAL REQMT CLASSROOM CONTACT HOURS	REQMT VS AVAIL RATIO	RESOURCE UTIL. RANGE MINIMUM MAXIMUM	THROUGHPUT CHANGE PER UNIT RESOURCES CHANGE	LIMITING VARIABLE
DORSEY	838	1Ø42	465	44	465.Ø	NO CHG	
					465.Ø	NO CHG	
HUNT	838	1Ø42	5Ø4	48	5Ø4.Ø	NO CHG	
					5Ø4.Ø	NO CHG	
STEWART	838	1Ø42	48Ø	46	48Ø.Ø	NO CHG	
					48Ø.Ø	NO CHG	
WARD	838	1Ø42	4Ø8	39	4Ø8.Ø	NO CHG	
					4Ø8.Ø	NO CHG	
BROWN	839	1Ø41	825	79	825.Ø	NO CHG	
					825.Ø	NO CHG	
PAUL	838	1Ø42	4Ø8	39	4Ø8.Ø	NO CHG	
					4Ø8.Ø	NO CHG	
STREIT	838	1Ø42	48Ø	46	48Ø.Ø	NO CHG	
					48Ø.Ø	NO CHG	
ROHLFING	838	1Ø42	48Ø	46	48Ø.Ø	NO CHG	
					48Ø.Ø	NO CHG	
BROOKS	838	1Ø42	585	56	585.Ø	NO CHG	
					585.Ø	NO CHG	
SPARKES	838	1Ø42	48Ø	46	48Ø.Ø	NO CHG	
					48Ø.Ø	NO CHG	
HARRISON	838	1Ø42	432	41	432.Ø	NO CHG	
					432.Ø	NO CHG	
OBERLE	878	1ØØ2	78Ø	77	78Ø.Ø	NO CHG	
					78Ø.Ø	NO CHG	
WAGNER	838	1Ø42	78Ø	74	78Ø.Ø	NO CHG	
					78Ø.Ø	NO CHG	
FLTLNCR		2Ø8Ø	288	13	288.Ø	NO CHG	
					288.Ø	NO CHG	
N-3Ø1Ø6		2Ø8Ø	198	9	198.Ø	NO CHG	
					198.Ø	NO CHG	
N-3Ø1Ø7		2Ø8Ø	822	39	822.Ø	NO CHG	
					822.Ø	NO CHG	
N-3Ø1Ø8		2Ø8Ø	24Ø	11	24Ø.Ø	NO CHG	
					24Ø.Ø	NO CHG	
N-3Ø167		2Ø8Ø	1ØØØ	48	1ØØØ.Ø	NO CHG	
					1ØØØ.Ø	NO CHG	
N-3Ø18Ø		2Ø8Ø	1Ø8Ø	51	1Ø8Ø.Ø	NO CHG	
					1Ø8Ø.Ø	NO CHG	
N-3Ø181		2Ø8Ø	5Ø4	24	5Ø4.Ø	NO CHG	
					5Ø4.Ø	NO CHG	
N19A12Ø		2Ø8Ø	96Ø	46	96Ø.Ø	NO CHG	
					96Ø.Ø	NO CHG	
N19A122		2Ø8Ø	48Ø	23	48Ø.Ø	NO CHG	
					48Ø.Ø	NO CHG	

f) To modify the availability of an instructor or a facility, type:

MODAVAIL

followed by a carriage return. The system will respond:

```
EXECUTION:
ENTER NAME OF INSTRUCTOR OR FACILITY:
```

Then type:

BROWN

followed by a carriage return. The system will respond:

```
ENTER CURRENT AVAILABLE HOURS, NEW AVAILABLE HOURS:
```

Then type:

1041 1250

followed by a carriage return. The system will respond:

```
AVAIL     27242371   1041    35032870   1042
AVAIL     27242371   1250    35032870   1042
```

to indicate that the availability change has been made. The SCRR OUTPUT shown in step c) indicates that the greatest improvement in total student throughput can be achieved by increasing BROWN's availability. Therefore, before generating a new SCRR OUTPUT listing, BROWN's availability is changed from the current 1041 hours to the indicated minimum requirement of 1250 hours.

The system will then ask:

```
ANY MORE CHANGES?  (YES/NO)
```

Type:

NO

followed by a carriage return. The system will respond:

```
CHANGES TO AVAILABLE HOURS HAVE BEEN INCORPORATED.
TO RERUN SCRR MODEL, TYPE:  SCRR1
```

As directed in the above message, to rerun the SCRR model type:

SCRR1

followed by a carriage return. The system will respond:

```
MPS III, ON-LINE VERSION, READY.
MPS COMMAND:
MPS III TERMINATING.
TO PRINT SCRR MODEL OUTPUT, TYPE:
P SCRR OUTPUT
```

To print the SCRR OUTPUT which now includes the availability change made above, type:

P SCRR OUTPUT

Test procedure f) is successful if the following output is duplicated exactly.

RESOURCE	ANNUAL NON-CLASSROOM HOURS	ANNUAL AVAIL CLASSROOM CONTACT HOURS	ANNUAL REQMT CLASSROOM CONTACT HOURS	REQMT VS AVAIL RATIO	RESOURCE UTIL. RANGE MINIMUM MAXIMUM	THROUGHPUT CHANGE PER UNIT RESOURCES CHANGE	LIMITING VARIABLE
DORSEY	838	1042	890	85	788.0	-0.492	510V
					890.0	NO CHG	
HUNT	838	1042	504	48	504.0	NO CHG	
					1504.0	-0.098	461V
STEWART	838	1042	1040	99	480.0	-0.400	N19A120
					1040.0	NO CHG	
WARD	838	1042	940	90	671.0	-0.196	461V
					1042.0	-0.492	510V
BROWN	839	1250	1250	100	1148.0	-0.725	PAUL
					1282.0	0.725	510G
PAUL	838	1042	940	90	671.0	-0.196	461V
					1042.0	-0.492	510V
STREIT	838	1042	1040	99	480.0	-0.400	N19A120
					1040.0	NO CHG	
ROHLFING	838	1042	1040	99	480.0	-0.400	N19A120
					1040.0	NO CHG	
BROOKS	838	1042	1042	100	1010.0	-0.350	510G
					1144.0	0.350	PAUL
SPARKES	838	1042	1040	99	480.0	-0.400	N19A120
					1040.0	NO CHG	
HARRISON	838	1042	1042	100	432.0	-0.167	510W
					3120.0	0.167	FLTLHGR
OBERLE	870	1092	1092	100	780.0	-0.100	5698
					1042.0	0.100	WAGNER
WAGNER	838	1042	1000	96	780.0	-0.100	OBERLE
					1000.0	NO CHG	
FLTLHGR		2080	695	33	288.0	-0.250	HARRISON
					694.7	NO CHG	
N-3010 6		2080	251	12	198.0	-0.212	BROOKS
					419.1	-0.298	510V
N-3010 7		2080	1053	50	865.2	-0.127	OBERLE
					1190.7	-0.364	510V
N-3010 8		2080	308	14	240.0	-0.350	OBERLE
					308.3	NO CHG	
N-3010 67		2080	1914	92	1694.6	-0.229	510V
					1914.0	NO CHG	
N-3010 80		2080	2080	100	1080.0	-0.146	536P
					2284.0	0.146	PAUL
N-3010 81		2080	707	34	504.0	-0.500	HARRISON
					809.3	-0.492	510V
N19A120		2080	2080	100	960.0	-0.200	5107
					2084.0	0.200	SPARKES
N19A122		2080	1040	50	480.0	-0.400	N19A120
					1040.0	NO CHG	

COURSE CDP NO.	MAXIMUM ANNUAL CONVEN	CURRENT SCHEDULED CONVEN	AVERAGE CLASS SIZE	ANNUAL CONVEN RANGE MINIMUM MAXIMUM	THROUGHPUT CHANGE PER COURSE CONVEN CHANGE	CLASS SIZE RANGE MINIMUM MAXIMUM	LIMITING VARIABLE
461V	12.Ø	12	2	Ø.Ø	-2.Ø	Ø.Ø	
				24.8	2.Ø	+INF	HUNT
51ØG	6.Ø	6	7	Ø.Ø	-7.Ø	Ø.Ø	
				28.8	7.Ø	+INF	BROOKS
51ØV	12.Ø	12	7	Ø.Ø	-7.Ø	Ø.Ø	
				26.2	7.Ø	+INF	BROWN
51ØW	24.Ø	24	3	Ø.Ø	-3.Ø	Ø.Ø	
				57.9	3.Ø	+INF	HARRISON
51Ø7	24.Ø	24	8	Ø.Ø	-8.Ø	Ø.Ø	
				52.Ø	8.Ø	+INF	N-19A12Ø
536K	5Ø.Ø	5Ø	1Ø	Ø.Ø	-1Ø.Ø	Ø.Ø	
				95.7	1Ø.Ø	+INF	BROWN
536P	12.Ø	12	7	Ø.Ø	-7.Ø	Ø.Ø	
				32.8	7.Ø	+INF	N-3Ø18Ø
5698	6.Ø	6	14	Ø.Ø	-14.Ø	Ø.Ø	
				7.7	14.Ø	+INF	OBERLE

Before proceeding to the next step, type:

MAXCONV

followed by a carriage return. Upon successful completion, the system will respond:

DATA MODIFIED TO OPTIMIZE COURSE CONVENINGS.
TO RERUN THE SCRR MODEL, TYPE: SCRR1

g) To initialize the requirement modification routine for the operational test, type:

TESTREQ

followed by a carriage return. Upon successful completion, the system will respond with the time-of-day.

h) To modify training system requirements, type:

REQMT

followed by a carriage return. The system will respond as follows:

DO YOU WISH INSTRUCTIONS TO BE PRINTED? (YES/NO):

Type:

YES

followed by a carriage return.

The system will respond:

```
THIS PROGRAM PROVIDES THE CAPABILITY TO MODIFY COURSE UTILIZATION,
CLASS CAPACITY, AND COURSE LENGTH EXPRESSED IN CONTACT HOURS TO
ASSESS THE IMPACT ON TOTAL INSTRUCTOR REQUIREMENTS.
*  TO INCREASE UTILIZATION 2Ø% ENTER: .2Ø
*  TO DECREASE UTILIZATION 1Ø% ENTER: -.1Ø
*  IF NO CHANGE IS DESIRED ENTER: Ø
ENTER UTILIZATION DELTA:
```

Type:

```
-.5
```

followed by a carriage return. The system will respond:

```
* TO INCREASE CLASS CAPACITIES 2Ø% ENTER: .2Ø
* TO DECREASE CLASS CAPACITIES 1Ø% ENTER: -.1Ø
* IF NO CHANGE IS DESIRED ENTER: Ø
ENTER CLASS CAPACITY DELTA:
```

Type:

```
0
```

followed by a carriage return. The system will respond:

```
* TO INCREASE COURSE LENGTH 2Ø% ENTER: .2Ø
* TO DECREASE COURSE LENGTH 1Ø% ENTER: -.1Ø
* IF NO CHANGE IS DESIRED ENTER:  Ø
ENTER COURSE LENGTH DELTA:
```

Type:

```
0
```

followed by a carriage return. Upon successful completion, the system will respond:

```
CHANGES HAVE BEEN INCORPORATED.
TO RERUN SCRR MODEL, TYPE:  SCRR2
```

To verify that the REQMT routine executed successfully, the user should now rerun the ASSIGN program (see test procedure step d)). Test procedure h) is successful if the following output is duplicated exactly.

INSTRUCTOR	COURSE	CLASIZE	CONV	HRS/CONV	TOT HRS	AVAIL HRS
BROOKS	51ØG	4	6	2Ø.Ø	12Ø.Ø	
	536K	5	5Ø	6.7	335.Ø	
				TOT =	455.Ø	1Ø42
BROWN	51ØV	3	12	3Ø.Ø	36Ø.Ø	
	536K	5	5Ø	6.7	335.Ø	
				TOT =	695.Ø	1Ø41
DORSEY	536K	5	5Ø	6.7	335.Ø	1Ø42
HARRISON	51ØW	1	24	18.Ø	432.Ø	1Ø42
HUNT	461V	1	12	42.Ø	5Ø4.Ø	1Ø42
OBERLE	5698	7	6	75.Ø	45Ø.Ø	1ØØ2
PAUL	51ØG	4	6	2Ø.Ø	12Ø.Ø	
	536P	4	12	24.Ø	288.Ø	
				TOT =	4Ø8.Ø	1Ø42
ROHLFING	51Ø7	4	24	15.Ø	36Ø.Ø	1Ø42
SPARKES	51Ø7	4	24	15.Ø	36Ø.Ø	1Ø42
STEWART	51Ø7	4	24	15.Ø	36Ø.Ø	1Ø42
STREIT	51Ø7	4	24	15.Ø	36Ø.Ø	1Ø42
WAGNER	5698	7	6	75.Ø	45Ø.Ø	1Ø42
WARD	51ØG	4	6	2Ø.Ø	12Ø.Ø	
	536P	4	12	24.Ø	288.Ø	
				TOT =	4Ø8.Ø	1Ø42

i) The final step of the SCRR operational test is to reset the SCRR software so that model input data are again computed from the DOTS data base files. This is accomplished by typing:

TESTOFF

followed by a carriage return. The system will respond with the time-of-day. Then type:

TESTREQO

followed by a carriage return. The system will respond with the time-of-day.

SCRR MODEL OPERATIONAL TEST PROCEDURE SUMMARY

Keyboard Entry	Operator Action Required
SCRRTEST	None
SET CORE 320	None
SCRR	Respond to Program Queries
P SCRR OUTPUT	Review Output for Accuracy
ASSIGN	None
P ASSIGN LISTING	Review Output for Accuracy
FIXCONV	None
SCRR1	None
P SCRR OUTPUT	Review Output for Accuracy
MAXCONV	None
MODAVAIL	Respond to Program Queries
SCRR1	None
P SCRR OUTPUT	Review Output for Accuracy
TESTREQ	None
REQMT	Respond to Program Queries
ASSIGN	None
P ASSIGN LISTING	Review Output for Accuracy
TESTOFF	None
TESTREQO	None

SECTION V

DOTS DATA BASE

PROGRAM DESCRIPTIONS

INTRODUCTION

Documentation relative to the software elements developed for the maintenance of the three files of the DOTS data base is included within this section. This includes software for the NCSS timesharing computer and the WANG computer system.

First, are the RAMIS data management system file descriptions and the executives established for their control. A description follows covering the types of maintenance activities available. Next, the WANG computer system software is discussed, followed by the NCSS software developed for DOTS data base maintenance.

The NCSS executives contained herein are termed "executive sequences," since most are several individual VP/CSS executives in sequence. Due to the interactive attributes of the DOTS system, executives are documented by reference to the root executive, or that to be invoked by the user at the terminal.

Program documentation contains a functional description of the task to be performed, input/output data formats (where applicable), and program source code listings.

Interaction of the user with the data base maintenance software on both the WANG and NCSS systems is covered in detail in SECTION II of this manual and is not, in general, reproduced in any detail within this section.

GENERAL SYSTEMS PROCEDURES

The implications associated with the following RAMIS file descriptions and VP/CSS executive sequences presupposes a rather detailed knowledge on the part of the user with basic services provided by the National CSS (NCSS) time-sharing vendor. These services are formally documented by NCSS and are not reproduced within this manual. To assist those who are unfamiliar with the NCSS documentation, the following references are mentioned:

System Function	NCSS Reference
Typing Conventions, Login, Logout	A Guide to the National CSS Computer System for Application Product Users (NCSS Form 991) Section 1
System Interruptions	A Guide to the National CSS Computer System for Application Product Users (NCSS Form 991) Section 7
CSS Commands Summary	A Guide to the National CSS Computer System for Application Product Users (NCSS Form 991) Appendix B
CSS File Creation and Maintenance	A Guide to the National CSS Computer System for Application Product Users (NCSS Form 991) Section 3
	VP/CSS Edit Command (NCSS Form 108)
Data Base Maintenance Functions Data Base Report Generation RAMIS System RUN PROCEDURES	RAMIS Users' Manual (Mathematica, 1976 Edition)
VP/CSS Executive Language	VP/CSS Executive Language (EXEC) (NCSS Form 109)

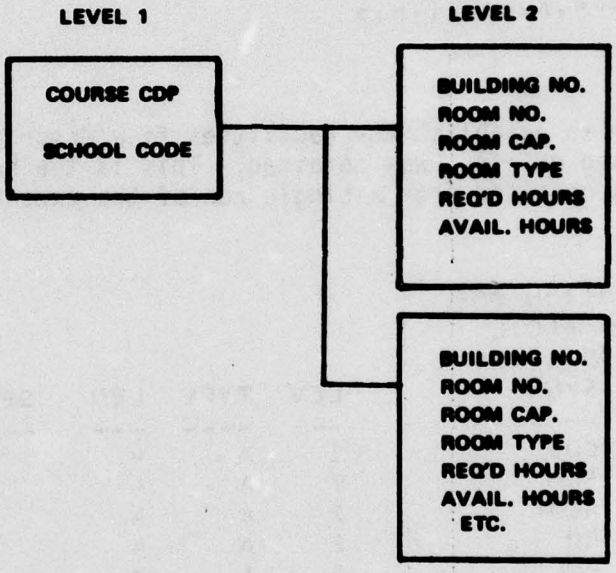
RAMIS DATA MANAGEMENT SYSTEM FILE DESCRIPTIONS

DOTS DATA BASE : Facilities

RAMIS NAME : FACIL

RAMIS DESCRIPTION : Filename = FACIL
Filetype = DESC

The tree-structure of the DOTS Facilities file is as depicted on the diagram following:



DOTS FACILITIES FILE – HIERARCHIC STRUCTURE

FIGURE V-1

The key field controlling the highest level segment is course CDP. The level 2 segment is keyed by building number.

The RAMIS file description which was used to establish and support the above structure is as follows. This is the NCSS card-image format.

Listing of Filename = FACIL, Filetype = DESC

```
ERASE*
FILE FACIL
END
INPUT
FILE RAMASTER
1=FACIL,1,S,12,CDP,CDP,A,4,$
   5=SCH,SCH,A,4,,
2=2,S,5,BLDG,BLDG,A,4,$
   5=RM,RM,A,4,,
   5=RMCAP,RMCAP,I,3,,
   5=RMTP,RMTP,A,1,$
   5=REQHRS,REQHRS,F,6.1,$
   5=AVHRS,AVHRS,I,5,,
END
```

After using the above to establish the Facilities file description internal to RAMIS, the following printout was obtained. This is the RAMIS listing of the file format as obtained from a single run of the executive FACIL with the INDEX option.

```
   PAMASTER DESCRIPTION FOR
   RAMIS FILE NAMED FACIL
   AT 15.48.35 ON 09/10/76
```

LIST	FIELD NAME	SYNONYM	LEV	TYPE	LEN	SEG	SFLAG
1	CDP	CDP	1	A	4	12	S
2	SCH	SCH	1	A	4	12	S
3	BLDG	BLDG	2	A	4	5	S
4	RM	RM	2	A	4	5	S
5	RMCAP	RMCAP	2	I	3	5	S
6	RMTP	RMTP	2	A	1	5	S
7	REQHRS	REQHRS	2	F	6.1	5	S
8	AVHRS	AVHRS	2	I	5	5	S

A RAMIS dump function produces a dump file in card-image format. This same 80-byte format is maintained, regardless of the internal RAMIS file record size. In the case of the Facilities file, a total of 33 bytes are required to contain a single internal record. Thus, the Facilities dump file will contain card images, using the first 33 bytes and blanking the remaining 47 bytes.

Execution of a RAMIS DUMP results in a RAMIS system printout of the format of the dump records. For the Facilities file, the following results:

```
 DUMP
FILE FACIL
PRINT AVHRS AND SAVE
IF SCH IS 6$$$
END
 EBCDIC RECORD LENGTH 8Ø CHARACTERS

 FIELD NAME      FORMAT        LENGTH

 CDP             A    4          4
 SCH             A    4          4
 BLDG            A    4          4
 PM              A    4          4
 RMCAP           I    3          3
 RMTP            A    1          1
 REQHRS          F6   .1         6
 AVHRS           I    5          5
   NUMBER OF RECORDS DUMPED=     188
```

The Facilities file data field contents and related maintenance details are itemized in the following table. Complete field definitions for the fields found in the Facilities file is covered in SECTION II of this manual.

FACILITIES FILE

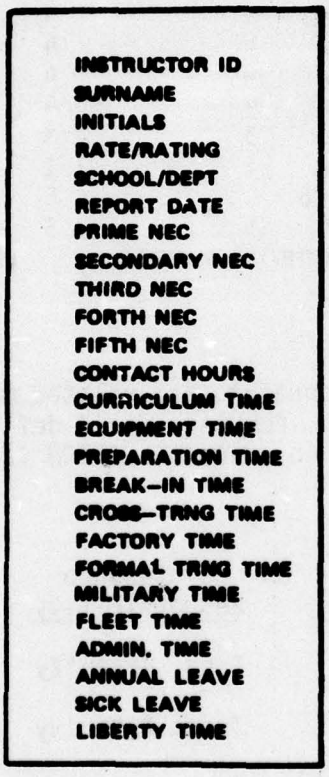
Field Name (Mnemonic), Data Type	Maintenance Responsibility	Notes
Course CDP Number (CDP), A4	Trng. Activity	From internal records
School Code (SCH), A4	Trng. Activity	From internal records
Building Number (BLDG), A4	Trng. Activity	From internal records
Room Number (RM), A4	Trng. Activity	From internal records
Room Capacity (RMCAP), I3	Trng. Activity	From internal records
Room Type (RMTP), A1 1 = Lecture; 2 = Lab; 3 = Both	Trng. Activity	From internal records
Required Hours (REQHRS), F6.1	Trng. Activity	From internal records
Available Hours (AVHRS), I5	Trng. Activity	From internal records

DOTS DATA BASE : Instructor

RAMIS NAME : INST

RAMIS DESCRIPTION : Filename = INST
Filetype = DESC

The following diagram indicates pictorially the hierarchic (tree) structure of the DOTS Instructor file:



DOTS INSTRUCTOR FILE – HIERARCHIC STRUCTURE

FIGURE V-2

The RAMIS description supporting the above structure follows:

Listing of Filename = INST, Filetype = DESC

```
ERASE*
FILE INST
END
INPUT
FILE RAMASTER
1=INST,1,5,100,ID,10,I,9,*
   5=NME,NME,A,16,$
   5=IN,IN,A,2,$
   5=RATE,RATE,A,5,$
   5=SCH,SCH,A,4,$
   5=REP,REP,I,4,$
   5=ROT,ROT,I,4,$
   5=PNEC,PNEC,A,4,*
   5=SNEC,SNEC,A,4,*
   5=NEC3,NEC3,A,4,*
   5=NEC4,NEC4,A,4,*
   5=NEC5,NEC5,A,4,*
   5=CONT,CONT,I,4,*
   5=CDEV,CDEV,I,4,*
   5=EQMT,EQMT,I,4,*
   5=PREP,PREP,I,4,*
   5=OKIN,OKIN,I,4,*
   5=CTRN,CTRN,I,4,*
   5=FTRN,FTRN,I,4,*
   5=MDUT,MDUT,I,4,*
   5=ISCH,ISCH,I,4,*
   5=FLET,FLET,I,4,*
   5=ADMN,ADMN,I,4,*
   5=ANLV,ANLV,I,4,*
   5=SKLV,SKLV,I,4,*
   5=SLIB,SLIB,I,4,*
END
```

The internal RAMIS format established as a result of the above description is shown below. This listing results from a run of executive INST with an INDEX option:

RAMASTER DESCRIPTION FOR
RAMIS FILE NAMED INST
AT 15.50.48 ON 09/10/76

LIST	FIELD NAME	SYNONYM	LEV	TYPE	LEN	SEG	SFLAG
1	ID	ID	1	I	9	100	S
2	NME	NME	1	A	16	100	S
3	IN	IN	1	A	2	100	S
4	RATE	RATE	1	A	5	100	S
5	SCH	SCH	1	A	4	100	S
6	REP	REP	1	I	4	100	S
7	ROT	ROT	1	I	4	100	S
8	PNEC	PNEC	1	A	4	100	S
9	SNEC	SNEC	1	A	4	100	S
10	NEC3	NEC3	1	A	4	100	S
11	NEC4	NEC4	1	A	4	100	S
12	NEC5	NEC5	1	A	4	100	S
13	CONT	CONT	1	I	4	100	S
14	CDEV	CDEV	1	I	4	100	S
15	EOMT	EOMT	1	I	4	100	S
16	PREP	PREP	1	I	4	100	S
17	BKIN	BKIN	1	I	4	100	S
18	CTRN	CTRN	1	I	4	100	S
19	ETRN	ETRN	1	I	4	100	S
20	MDUT	MDUT	1	I	4	100	S
21	ISCH	ISCH	1	I	4	100	S
22	FLET	FLET	1	I	4	100	S
23	ADMN	ADMN	1	I	4	100	S
24	ANLV	ANLV	1	I	4	100	S
25	SKLV	SKLV	1	I	4	100	S
26	SLIB	SLIB	1	I	4	100	S

The dump file formed by a RAMIS DUMP function directed at the Instructor file will contain two card-image dump records per internal RAMIS record. As seen from the following listing, the first of the card-image pairs is completely used (no padding required). Only 40 bytes are required on the second of the card-image pair to complete the internal record dump.

```
DUMP
FILE INST
PRINT SLIB AND SAVE
IF SCH IS 6$$$
END

EBCDIC RECORD LENGTH 80 CHARACTERS

FIELD NAME      FORMAT      LENGTH

ID              I   9           9
NME             A   16         16
IN              A   2           2
RATE            A   5           5
SCH             A   4           4
REP             I   4           4
ROT             I   4           4
PNEC            A   4           4
SNEC            A   4           4
NEC3            A   4           4
```

INSTRUCTOR FILE DUMP (Cont'd)

NEC4	A	4		4
NEC5	A	4		4
CONT	I	4		4
CDEV	I	4		4
EQMT	I	4		4
PREP	I	4		4
			X	Ø
BKIN	I	4		4
CTRN	I	4		4
ETRN	I	4		4
MDUT	I	4		4
ISCH	I	4		4
FLET	I	4		4
ADMN	I	4		4
ANLY	I	4		4
SKLV	I	4		4
SLIB	I	4		4

NUMBER OF RECORDS DUMPED= 211

The following table indicates the contents of the Instructor file data fields and the related maintenance responsibility. Complete details on the meanings and usage of these fields was outlined in Section II of this manual and will not be duplicated in this section.

INSTRUCTOR FILE

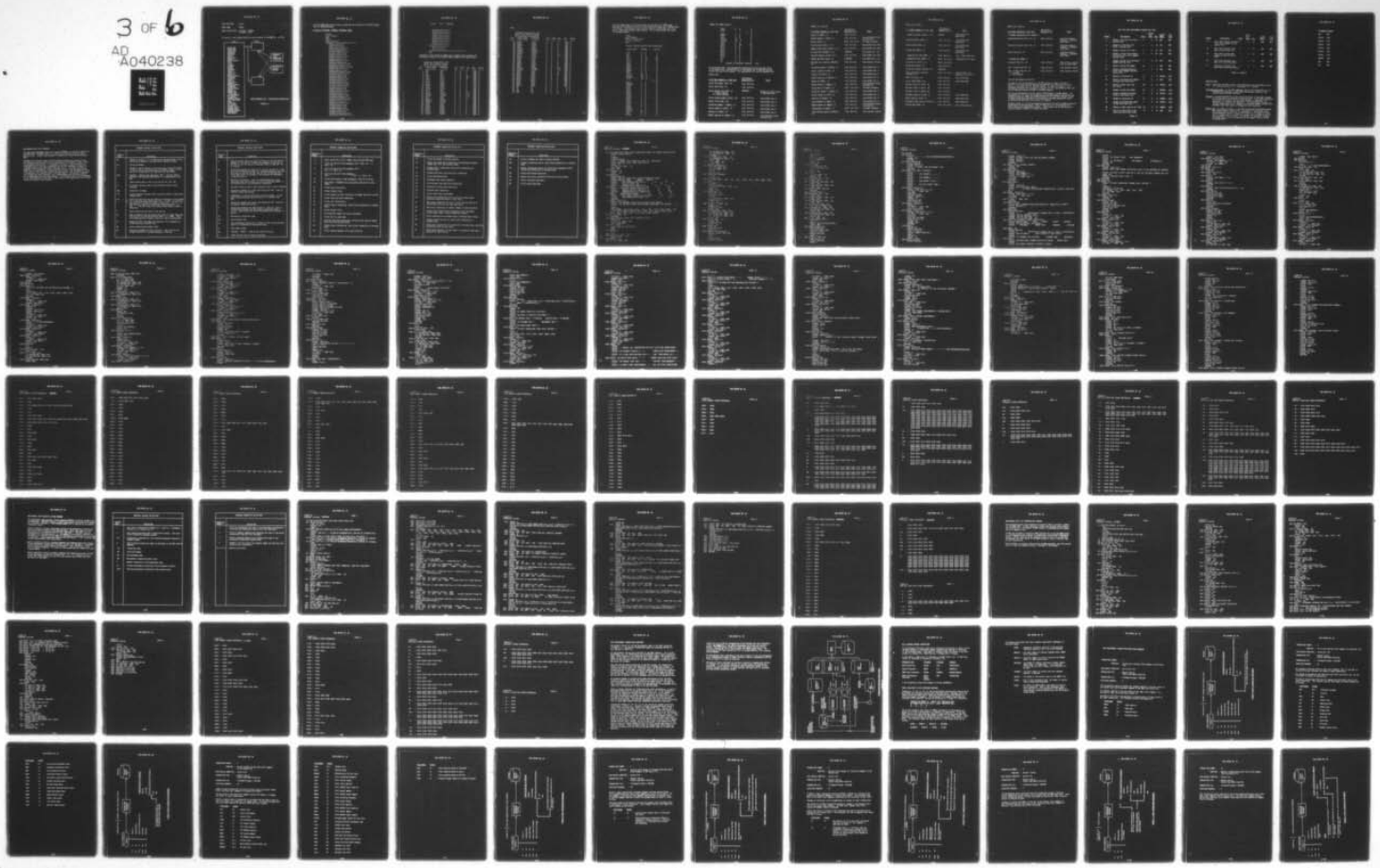
Field Name (Mnemonic), Data Type	Maintenance Responsibility	Notes
Instructor Social Security Number (ID), A9	Trng. Act. Pers. Dept.	From BUPERS reports
Instructor Surname (NME), A16	Trng. Act. Pers. Dept.	From BUPERS reports
Instructor Initials (IN), A2	Trng. Act. Pers. Dept.	From BUPERS reports
Rate/Rating (RATE), A5	Trng. Act. Pers. Dept.	From BUPERS reports
School (SCH), A4	Activity	
Reporting Date (REP), A4, YYMM	Trng. Act. Pers. Dept.	From BUPERS reports
Rotation Date (ROT), A4, YYMM	Trng. Act. Pers. Dept.	From BUPERS reports
Primary NEC (PNEC), A4	Trng. Act. Pers. Dept.	From BUPERS reports
Secondary NEC (SNEC), A4	Trng. Act. Pers. Dept.	From BUPERS reports

INSTRUCTOR FILE (Cont'd)

Field Name (Mnemonic), Data Type	Maintenance Responsibility	Notes
Third NEC (NEC3), A4	Trng. Act. Pers. Dept.	From BUPERS reports
Fourth NEC (NEC4), A4	Trng. Act. Pers. Dept.	From BUPERS reports
Fifth NEC (NEC5), A4	Trng. Act. Pers. Dept.	From BUPERS reports
Contact Hours (CONT), I4	Trng. Activity	From internal records
Curriculum Time (CDEV), I4	Trng. Activity	From internal records
Equipment Time (EQMT), I4	Trng. Activity	From internal records
Preparation Time (PREP), I4	Trng. Activity	From internal records
Breakin Time (BKIN), I4	Trng. Activity	From internal records
Cross-Training Time (CTRN), I4	Trng. Activity	From internal records
Factory Time (FTRN), I4	Trng. Activity	From internal records
Formal Training Time (ISCH), I4	Trng. Activity	From internal records
Military Time (MDUT), I4	Trng. Activity	From internal records
Fleet Training Time (FLET), I4	Trng. Activity	From internal records
Administrative Time (ADMN), I4	Trng. Activity	From internal records
Annual Leave (ANLV), I4	Trng. Activity	From internal records
Sick Leave (SKLV), Ir	Trng. Activity	From internal records
Special Liberty Time (SLIB), I4	Trng. Activity	From internal records

AD-A040 238 IBM CORP CAPE CANAVERAL FLA F/G 5/9
DESIGN OF TRAINING SYSTEMS. TRAPAC USER'S MANUAL, DATA BASE, SC--ETC(U)
SEP 76 K V BRANCH, J D STALEY, R E YANKO N61339-73-C-0097
UNCLASSIFIED TAEG-36 NL

3 OF 6
AD A040238

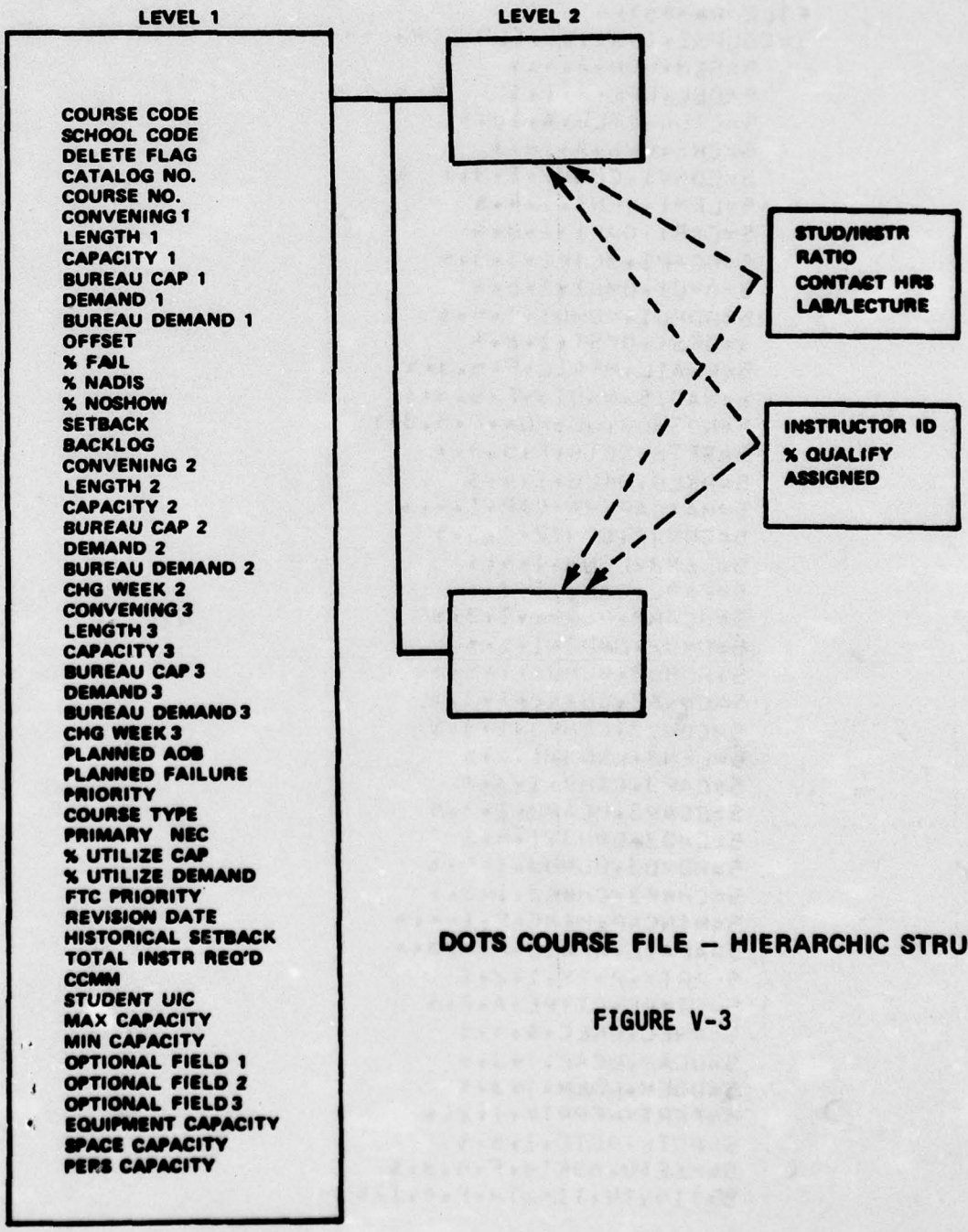


DOTS DATA BASE : Course

RAMIS NAME : COURSE

RAMIS DESCRIPTION : Filename = COURSE
Filetype = DESC

Following is the diagram depicting the structure of the DOTS Course file:



DOTS COURSE FILE – HIERARCHIC STRUCTURE

FIGURE V-3

The card-image description used to establish the structure of the DOTS Course file is indicated below:

Listing of Filename = COURSE, Filetype = DESC

```
ERASE*
FILE COURSE
END
INPUT
FILE RAMASTER
1=COURSE,1,S,100,CDP,CDP,A,4,$
   5=SCH,SCH,A,4,$
   5=DEL,DEL,A,1,$
   5=CTLG,CTLG,A,10,$
   5=CRS,CRS,A,16,$
   5=CONV1,CONV1,I,3,$
   5=LEN1,LEN1,I,4,$
   5=CAP1,CAP1,I,3,$
   5=BCAP1,BCAP1,I,3,$
   5=DMD1,DMD1,I,5,$
   5=RDMD1,RDMD1,I,5,$
   5=OFST,OFST,I,2,$
   5=PFAIL,PFAIL,F,5.3,$
   5=NADIS,NADIS,F,5.3,$
   5=NOSHOW,NOSHOW,F,5.3,$
   5=SETH,SETH,F,5.3,$
   5=BKLG,BKLG,I,4,$
   5=MAXCAP,MAXCAP,I,4,$
   5=CONV2,CONV2,I,3,$
   5=LEN2,LEN2,I,4,$
   5=CAP2,CAP2,I,3,$
   5=BCAP2,BCAP2,I,3,$
   5=DMD2,DMD2,I,5,$
   5=RDMD2,RDMD2,I,5,$
   5=CHWK2,CHWK2,I,3,$
   5=CONV3,CONV3,I,3,$
   5=LEN3,LEN3,I,4,$
   5=CAP3,CAP3,I,3,$
   5=BCAP3,BCAP3,I,3,$
   5=DMD3,DMD3,I,5,$
   5=RDMD3,RDMD3,I,5,$
   5=CHWK3,CHWK3,I,3,$
   5=MINCAP,MINCAP,I,4,$
   5=AFAIL,AFAIL,F,5.3,$
   5=PRTY,PRTY,I,2,$
   5=CTYPE,CTYPE,A,2,$
   5=PNEC,PNEC,A,4,$
   5=UCAP,UCAP,I,3,$
   5=UDEM,UDEM,I,3,$
   5=FPRTY,FPRTY,I,2,$
   5=RDTE,RDTE,I,6,$
   5=RSETH,RSETH,F,5.3,$
   5=TINSTR,TINSTR,F,4.1,$
```

COURSE DESC (Cont'd)

```
   5=SUIC,SUIC,A,5,$
   5=CCMM,CCMM,A,10,$
   5=OPT1,OPT1,A,6,$
   5=OPT2,OPT2,A,6,$
   5=OPT3,OPT3,A,6,$
   5=ECAP,ECAP,I,3,$
   5=SCAP,SCAP,I,3,$
   5=PCAP,PCAP,I,3,$
   5=PAOR,PAOR,I,4,$
2=2,S,5,F1,F1,I,4,$
   5=F2,F2,I,4,$
   5=F3,F3,I,1,$
2=3,V,1,NME,NME,A,16,,F1,INST,,1 1,$
END
```

Execution of the executive COURSE with an INDEX option produces the following listing of the internal RAMIS format of the Course file:

RAMASTER DESCRIPTION FOR
RAMIS FILE NAMED COURSE
AT 15.54.48 ON 09/10/76

LIST	FIELD NAME	SYNONYM	LEV	TYPE	LEN	SEG	SFLAG
1	CDP	CDP	1	A	4	100	S
2	SCH	SCH	1	A	4	100	S
3	DEL	DEL	1	A	1	100	S
4	CTLG	CTLG	1	A	10	100	S
5	CRS	CRS	1	A	16	100	S
6	CONV1	CONV1	1	I	3	100	S
7	LEN1	LEN1	1	I	4	100	S
8	CAP1	CAP1	1	I	3	100	S
9	BCAP1	BCAP1	1	I	3	100	S
10	DMD1	DMD1	1	I	5	100	S
11	BDMD1	BDMD1	1	I	5	100	S
12	OFST	OFST	1	I	2	100	S
13	PFAIL	PFAIL	1	F	5.3	100	S
14	NADIS	NADIS	1	F	5.3	100	S
15	NOSHOW	NOSHOW	1	F	5.3	100	S
16	SETB	SETB	1	F	5.3	100	S
17	BKLG	BKLG	1	I	4	100	S
18	MAXCAP	MAXCAP	1	I	4	100	S
19	CONV2	CONV2	1	I	3	100	S
20	LEN2	LEN2	1	I	4	100	S
21	CAP2	CAP2	1	I	3	100	S
22	BCAP2	BCAP2	1	I	3	100	S
23	DMD2	DMD2	1	I	5	100	S
24	BDMD2	BDMD2	1	I	5	100	S
25	CHWK2	CHWK2	1	I	3	100	S
26	CONV3	CONV3	1	I	3	100	S

PAGE 2

RAMASTER DESCRIPTION FOR
RAMIS FILE NAMED COURSE
AT 15.54.48 ON 09/10/76

LIST	FIELD NAME	SYNONYM	LEV	TYPE	LEN	SEG	SFLAG
27	LEN3	LEN3	1	I	4	100	S
28	CAP3	CAP3	1	I	3	100	S
29	BCAP3	BCAP3	1	I	3	100	S
30	DMD3	DMD3	1	I	5	100	S
31	BDMD3	BDMD3	1	I	5	100	S
32	CHWK3	CHWK3	1	I	3	100	S
33	MINCAP	MINCAP	1	I	4	100	S
34	AFAIL	AFAIL	1	F	5.3	100	S
35	PRTY	PRTY	1	I	2	100	S
36	CTYPE	CTYPE	1	A	2	100	S
37	PNEC	PNEC	1	A	4	100	S
38	UCAP	UCAP	1	I	3	100	S
39	UDEM	UDEM	1	I	3	100	S
40	FPRTY	FPRTY	1	I	2	100	S
41	RDTE	RDTE	1	I	6	100	S
42	HSETB	HSETB	1	F	5.3	100	S
43	TINSTR	TINSTR	1	F	4.1	100	S
44	SUIC	SUIC	1	A	5	100	S
45	CCMM	CCMM	1	A	10	100	S
46	OPT1	OPT1	1	A	6	100	S
47	OPT2	OPT2	1	A	6	100	S
48	OPT3	OPT3	1	A	6	100	S
49	ECAP	ECAP	1	I	3	100	S
50	SCAP	SCAP	1	I	3	100	S
51	PCAP	PCAP	1	I	3	100	S
52	PAOB	PAOB	1	I	4	100	S
53	F1	F1	2	I	9	5	S
54	F2	F2	2	I	4	5	S
55	F3	F3	2	I	1	5	S
56	NME	NME	3	A	16	1	V

The card-image dump file resulting from the execution of a RAMIS dump directed at the Course file will contain three card images per internal RAMIS file record. The internal format is such that all three cards are filled exactly (80 bytes) without blank padding. The following shows the layout of the dump records:

```
 DUMP
FILE COURSE
PRINT F3 AND SAVE
IF SCH IS 6$$$
END
```

EBCDIC RECORD LENGTH 8Ø CHARACTERS

FIELD NAME	FORMAT			LENGTH
CDP	A	4		4
SCH	A	4		4
DEL	A	1		1
CTLG	A	1Ø		1Ø
CRS	A	16		16
CONV1	I	3		3
LEN1	I	4		4
CAP1	I	3		3
BCAP1	I	3		3
DMD1	I	5		5
BDMD1	I	5		5
OFST	I	2		2
PFAIL	F5	.3		5
NADIS	F5	.3		5
NOSHOW	F5	.3		5
SETB	F5	.3		5
			X	Ø
BKLG	I	4		4
MAXCAP	I	4		4
CONV2	I	3		3
LEN2	I	4		4
CAP2	I	3		3
BCAP2	I	3		3
DMD2	I	5		5
BDMD2	I	5		5
CHWK2	I	3		3
CONV3	I	3		3
LEN3	I	4		4
CAP3	I	3		3
BCAP3	I	3		3
DMD3	I	5		5
BDMD3	I	5		5
CHWK3	I	3		3
MINCAP	I	4		4
AFAIL	F5	.3		5
PRTY	I	2		2
CTYPE	A	2		2

COURSE FILE DUMP (Cont'd)

PNEC	A	4		4
UCAP	I	3		3
			X	Ø
UDEM	I	3		3
FPRTY	I	2		2
RDTE	I	6		6
HSETB	F5	.3		5
TINSTR	F4	.1		4
SUIC	A	5		5
CCMM	A	1Ø		1Ø
OPT1	A	6		6
OPT2	A	6		6
OPT3	A	6		6
ECAP	I	3		3
SCAP	I	3		3
PCAP	I	3		3
PAOB	I	4		4
F1	I	9		9
F2	I	4		4
F3	I	1		1

NUMBER OF RECORDS DUMPED= 611

The following table links maintenance responsibility with the data fields of the Course file. See SECTION II of this manual for a detailed explanation of the data field content. This explanation will not be repeated here.

COURSE FILE

Field Name (Mnemonic), Data Type	Maintenance Responsibility	Notes
Course CDP Number (CDP), A4	Trng. Activity	
School Code (SCH), A4	Trng. Activity	
Course Delete Flag (DEL), A1 1 = Course Active 0 = Course Deleted	COMTRAPAC	Change to 0 when course is disestablished
Course Catalog Number (CTLG), A10	Trng. Activity	From NITRAS Form 17
Course Title (CRS), A16	Trng. Activity	From NITRAS Form 17
Convenings Number 1 (CONV1), I3	Trng. Activity	From NITRAS Form 17
Course Length #1 (LEN1), I4	Trng. Activity	From NITRAS Form 17
Capacity #1 (CAP1), I3	Trng. Activity	From NITRAS Form 17
BUPERS Capacity #1 (BCAP1), I3	Trng. Activity	From CNTECHTRA Annual Training Plan

COURSE FILE (Cont'd)

Field Name (Mnemonic), Data Type	Maintenance Responsibility	Notes
Demand #1 (DMD1), I5	Trng. Activity	
BUPERS Demand #1 (BDMD1), I5	Trng. Activity	From CNTECHTRA Annual Training Plan
Offset (OFST), I2	Maint. not req'd	TPF Model Parameter
Failure Rate (PFAIL), F5.3	Trng. Activity	From NITRAS Form 1204
NONAC Disenrollment (NADIS), F5.3	Trng. Activity	From NITRAS Form 1204
No-Show Rate (NOSHOW), F5.3	COMTRAPAC	From Quota Cont. Sys.
Percent Setbacks (SETB), F5.3	Trng. Activity	From NITRAS Form 1204
Number Backlogs (BKLG), I4	COMTRAPAC	From Quota Cont. Sys.
Maximum Class Capacity (MAXCAP), I4	Trng. Activity	From internal records
Convenings Number 2 (CONV2), I3	Trng. Activity	From NITRAS Form 17
Course Length #2 (LEN2), I4	Trng. Activity	From NITRAS Form 17
Capacity #2 (CAP2), I3	Trng. Activity	From NITRAS Form 17
BUPERS Capacity #2 (BCAP2), I3	Trng. Activity	From NITRAS Form 17
Demand #2 (DMD2), I5	Trng. Activity	From NITRAS Form 17
BUPERS Demand #2 (BDMD2), I5	Trng. Activity	From CNTECHTRA Annual Training Plan
Change Week #2 (CHWK2), I3	Maint. not req'd	TPF Model Parameter
Convenings Number 3 (CONV3), I3	Trng. Activity	From NITRAS Form 17
Course Length #3 (LEN3), I4	Trng. Activity	From NITRAS Form 17
Capacity #3 (BCAP3), I3	Trng. Activity	From NITRAS Form 17
Course Demand #3 (DMD3), I5	Trng. Activity	From NITRAS Form 17
BUPERS Demand #3 (BDMD3), I5	Trng. Activity	From CNTECHTRA Annual Training Plan
Change Week #3 (CHWK3), I3	Maint. not req'd	TPF Model Parameter
Class Minimum Capacity (MINCAP), I4	Trng. Activity	From internal records

COURSE FILE (Cont'd)

Field Name (Mnemonic), Data Type	Maintenance Responsibility	Notes
Academic Failures (AFAIL), F5.3	COMTRAPAC	From NITRAS MCRF (quarterly)
Course Priority (PRTY), I2	Trng. Activity	From NITRAS MCRF (quarterly)
Course Type (CTYPE), A2	Trng. Activity	From NITRAS MCRF (quarterly)
Primary NEC (PNEC), A4	Trng. Activity	From NITRAS MCRF (quarterly)
% Capacity Utilized (UCAP), I3	Trng. Activity	Calc. from NITRAS data
% Demand Utilized (UDEM), I3	Maint. not req'd	Generated by TPF model
FTC Priority (FPRTY), I2	Maint. not req'd	
Review Date (RDTE), I6	Trng. Activity	
Historical Setback (HSETB), F5.3	COMTRAPAC	
Total Instructors Required (TINSTR), F4.1	Maint. not req'd	From CNTECHTRA Inst., Computation Form 5311-1
Staff UIC (SUIC), A5	Trng. Activity	From NITRAS MCRF
Course CCMM (CCMM), A10	Trng. Activity	From NITRAS MCRF
Optional Field #1 (OPTI), A6	Trng. Activity	
Optional Field #2 (OPT2), A6	Trng. Activity	
Optional Field #3 (OPT3), A6	Trng. Activity	
Equipment Capacity (ECAP), I3	Trng. Activity	From NITRAS MCRF
Space Capacity (SCAP), I3	Trng. Activity	From NITRAS MCRF
Personnel Class Capacity (PCAP), I3	Trng. Activity	From NITRAS MCRF
Planned AOB (PAOB), I4	Maint. not req'd	Calc. from data base

COURSE FILE (Cont'd)

Field Name (Mnemonic), Data Type	Maintenance Responsibility	Notes
** STUDENT/INSTRUCTOR RATIO SEGMENT **		
Student/Instructor Ratio (F1), I9	Trng. Activity	From Course Master Schedule - CNTECHTRA Form 1540-1
Instructor Contact Hours (F2), I4	Trng. Activity	From Course Master Schedule - CNTECHTRA Form 1540-1
Course Type (F3), A1	Trng. Activity	From Course Master Schedule - CNTECHTRA Form 1540-1
** INSTRUCTOR SEGMENT **		
Instructor SSN (F1), I9	Trng. Activity	From internal records (list all qualified)
Instr. % Qualified (F2), I4	Trng. Activity	From internal records
Instr. Assign Flag (F3), A1 1 = assigned 0 = not assigned	Trng. Activity	From internal records

TYPES OF MAINTENANCE ACTIVITIES

The general flow of maintenance in the DOTS system is shown in Figure I-1. Table V-1 lists the distinct types of maintenance activities that can be directed at the three files of the DOTS system. As shown in Figure I-1, all maintenance transactions originate at the user site as constructed by the Transaction Prompting Program running in the WANG computer.

Each unique maintenance activity type is identified by a format number. All transactions of all types are accumulated at the WANG computer and sorted into ascending order by format type before transmssion to the NCSS computer system. A single transaction tape may be constructed on the WANG system with single or multiple transactions of any one or all formats.

Following Table V-1 are the programming descriptions of both the WANG processing program and the NCSS processor. Much of Table V-1 will be meaningful only after reading the program descriptions. A preliminary key to the meaning of certain columns in Table V-1 follows the table.

DOTS DATA BASE MAINTENANCE TRANSACTION TYPES

Format	Description	Files	Sing./ Mult. Step	Seq.	Output Name	File Type
2	Change a Facility File level 2 non-key field	F	S	17	DSC	RAM
3	Change an Instructor file level 1 non-key field	I	S	18	DSC	RAM
6	Change a Course file level 1	C	S	19	DSC	RAM
7	Change a Course file student/ instructor ratio (SIR) segment non-key field	C	S	20	DSC	RAM
8	Change a Course file instructor segment non-key field	C	S	21	DSC	RAM
11	Delete a course CDP number	FC	S	1	DSA	RAM
12	Delete a student/instructor ratio (SIR) segment from the Course file	C	S	2	DSA	RAM
13	Delete an instructor ID	IC	M	5	M13EXA	EXEC
14	Delete a building/room segment from the Facilities file	F	S	3	DSA	RAM
15	Delete a school code from all files	FIC	M	6	M15EXA	EXEC
16	Change a course CDP number	FC	M	7	M16EXA	EXEC
17	Change a student/instructor ratio in the Course file	C	M	8	M17EXA	EXEC
18	Change an instructor ID	IC	M	9	M18EXA	EXEC
19	Change a building/room number in the Facilities file	F	M	10	M19EXA	EXEC
20	Change a school code in all files	FIC	M	11	M20EXA	EXEC
21	Add a new course CDP to the Course and Facilities files	FC	S	12	DSB	RAM

TABLE V-1

Format	Description	Files	Sing./ Mult. Step	Seq.	Output Name	File Type
23	Add a new student/instructor ratio (SIR) segment to the Course file	C	S	13	DSB	RAM
24	Add a new instructor segment to the Course file	C	S	14	DSB	RAM
25	Add a new building/room segment to the Facilities file	F	S	15	DSB	RAM
26	Add a new instructor segment to an existing course	C	S	16	DSB	RAM
27	Delete an instructor segment from an existing course	C	S	4	DSA	RAM

TABLE V-1 (Cont'd)

Table V-1 Key

Files - Identifies the DOTS file(s) to be modified by the maintenance action. "F" = Facilities file; "I" - Instructor; "C" = Course.

Single/Multiple Step - At the NCSS computer, some of the transactions can be processed by executing a single RAMIS maintenance function; some require multiple RAMIS functions.

Sequence - A given maintenance run may include additions of new data, changes of existing data, or deletion of existing data. Since some maintenance activities are related, the following sequence is mandatory: all deletions, all changes, all additions. If a maintenance run were to include transactions of all formats, the sequence number indicates the absolute order that would be followed in their execution.

Output File - As shown in Figure I-1, the NCSS Transaction Processing Program may produce several output files in response to transaction input. The construction of each file is conditional; if the specific transaction format(s) that is related to a given file is absent in the input, no output file is written. To maintain the execution sequence discussed above, use of the contents of the output files is in the following order:

Filename	Filetype
DSA	RAM
M13EXA	EXEC
M15EXA	EXEC
M16EXA	EXEC
M17EXA	EXEC
M18EXA	EXEC
M19EXA	EXEC
M20EXA	EXEC
DSB	RAM
DSC	RAM

MAINTENANCE DATA ENTRY PROGRAM

The data base maintenance data entry program DBPROMPT is written in BASIC for the WANG 2200. DBPROMPT requires a single tape cassette drive and 16K of memory to run. A blank tape cassette is required to record the maintenance data.

DBPROMPT first requests that a data cassette be mounted and then positions the cassette tape to record the first maintenance transaction. Next, a menu of maintenance transactions is presented to the user. Upon selection of a maintenance action the user is prompted for the data necessary to make the indicated change in the NCSS RAMIS data base files. The data is stored in segments of an alphanumeric string variable until all data has been entered. After the last required data element has been entered, the alphanumeric string is recorded on the cassette tape in 100 byte blocks. This cycle is repeated until either the data cassette is full (approximately 650 100 byte blocks) or all maintenance data has been entered. If additional data is to be appended to the data cassette at some time in the future, a 256 type end-of-file trailer record is written on the cassette.

DBPROMPT VARIABLE DEFINITIONS

VARIABLE NAME	DEFINITION
C$	Character (usually "-") to indicate the maximum number of characters to be entered in the data entry subroutine (105).
C2$	Course CDP Number.
E	Automatic return indicator in the data entry subroutine (105). If E=0 sub 105 is not exited until a RETURN is keyed. If E=1 sub 105 is exited after last character has been keyed.
G$()	Constant. Contains the characters "END." The last record written on the cassette tape before it is transmitted to the DBM.
I	Index counter used in subs 0, 20, 59, 60, 91, 102, 105.
I$	A one byte variable used to store character data in keyin routines.
IØ$	Instructor ID number.
I1$	Single character variable used to key edit checks in data entry routine (105).
I9$	Identifies both the type and number of characters to be prompted for in the data entry subroutine (105). If I98 = "#" then the edit check will accept only numeric characters. If I98 = " a " then the edit check will accept only alpha characters. If I98 = " " the edit check will accept either alpha or numeric characters.
J	Index counter used in subs 31, 85, and 105.
J$	Used to construct the data base field name for fields that have FY+1 and FY+2 as part of the field name, i.e., CONV2. These field names may be included in type 6 or type 21 transactions.
J9	Length of I9$ or the number of characters to be prompted for by the data entry routine (105).
K$	Course Identification Number (CIN).
M$	Subroutine argument for subs 33 and 35. Used to print the variable portion of a standard message to the user.

DBPROMPT VARIABLE DEFINITIONS

VARIABLE NAME	DEFINITION
N	Used to specify block-size equal 100 bytes in the DATA SAVE BT commands. Also used as a subroutine argument in subs 30, 31, 32, 36, 42, 44, 55, and 56 to position cursor on desired line of display.
N$	Subroutine argument for subs 30, 33, 34, 35, 40, and 72. Used to print the variable portion of a standard message to the user. Also used in subs 39, 43, 88, 100, and 101 as the data base field name, which is included in the transaction alphanumeric string variable +$.
N1$	Multiple use variable. Used for student/instructor ratio, instructor name and initials, and for building/room number in the construction of the maintenance transaction.
Q$	One byte variable used to store character data in keyin routines.
Q1	Subroutine argument for cursor positioning sub 102. Used for vertical cursor positioning.
Q1$	Alphanumeric string variable used in keyin routines. If the keyin character passes the edit checks it is entered into Q1$.
Q5	Subroutine argument for cursor positioning sub 102. Used for horizontal cursor positioning.
R	Subroutine arguments for subs 38 and 40. Used as a flag to differentiate between add and changes transaction types in sub 38. Used to position cursor on desired line of display in sub 40.
R$	Maintenance transaction type.
S1$	Activity/dept code.
T$	Each maintenance transaction is stored in this string variable until it is recorded on the cassette tape.
W	Loop index counter.
W$	Constant = "AJKSX." Used to edit check CIN entry.
Y	Input variable used to control branching.

DBPROMPT SUBROUTINE DEFINITIONS	
SUBROUTINE NUMBER	DEFINITION
0	Abort subroutine; prints "ABORT" then displays MENU page.
3	Calls sub 105 with fixed arguments I98 = "###," C8 = "-," and E = Ø.
4	Calls sub 105 with fixed arguments I98 = " ," C8 = "-," and E = Ø.
5	Calls sub 105 with fixed arguments I98 = " ," C$ = "-," and E = Ø.
19	Prints eight dashes, eight backspaces, then calls sub 60.
20	Date (form = YY/MM/DD) entry subroutine with built-in edit checks.
21	Prints user instructions.
30	Prints display title.
31	Prints request to enter instructor ID number then calls sub 59.
32	Assign flag data entry subroutine.
33	Prints user instructions.
34	Creates type 8 transaction, then writes transaction on cassette tape.
35	Prints display title.
36	Building/room number data entry subroutine.
37	Prints facility data menu.
38	Facility data entry subroutine; differentiates between change data (R=1) and new data (R=Ø).
39	Creates type 2 transaction; then writes transaction on cassette tape.
40	Prints display heading; calls subs 55 and 56.

DBPROMPT SUBROUTINE DEFINITIONS	
SUBROUTINE NUMBER	DEFINITION
41	Prints CDP Number in display heading.
42	Contact hour data entry subroutine; differentiates between change data (N=1) and new data (N=Ø).
43	Creates type 7 transaction; then writes transaction on cassette tape.
44	Student/instructor ratio data entry subroutine.
50	Prints maintenance menu.
55	CDP Number data entry subroutine.
56	Activity/dept code data entry subroutine.
59	Instructor ID data entry subroutine.
60	CIN data entry subroutine.
72	Prints course data field names.
80	Prints user instructions for first half of course data; specifies subroutine branch on user reply.
81	Data entry subroutine for first 3 lines (18 data fields) of course data; calls sub 101 upon data entry.
85	Data entry subroutine for decimal numbers in the form NN.N%.
86	Creates first three fields (transaction type, CDP Number, activity/dept code) of every transaction record.
87	Writes transaction on cassette tape in 100 byte block format.
88	Creates transaction type 3, then writes transaction to cassette tape.
90	Prints user instructions for second half of course data; specifies subroutine branch on user reply.
91	Prints keyin character if edit check is successful; adds keyin character to keyin string.

DBPROMPT SUBROUTINE DEFINITIONS	
SUBROUTINE NUMBER	DEFINITION
92	Prints ID Number and name in display heading.
100	Creates transaction type 6, then writes transaction to cassette tape.
101	Creates transaction type 6 for fiscal year dependent fields, then writes transaction on cassette tape.
102	Cursor positioning subroutine.
105	General data entry subroutine with built-in edit checks.
124	Prints instructor data menu.
222	Prints course data menu.

PROGRAM LISTING - **DBPROMPT**

```
 10 DIM C2$4,S1$4,Q1$20,T$40,IO$9,N1$18,N$30,I$1,I9$20,I1$1,G$(1)50
     :SELECT PRINT 005(64)
     :0=0
     :REWIND
     :PRINT HEX(03)
     :STOP "; MOUNT DATA CASSETTE; THEN KEY CONTINUE."
     :PRINT HEX(03),"START NEW TAPE?(Y/N)"
     :INPUT I$
     :IF I$<>"N" THEN   40
     :SKIP END
 40 GOSUB '50
     :GOSUB   80
     :GOTO   40
 50 DEFFN'50
     :PRINT HEX(03)
     :PRINT TAB(17);"DOTS DATA BASE MAINTENANCE MENU"
     :PRINT TAB(29);"ADD       CHANGE    DELETE"
     :PRINT "    CDP NUMBER";TAB(30);"21          16          11"
     :PRINT "    ACTIVITY/DEPT NO.            *          20          *"
     :PRINT "    COURSE DATA";TAB(31);"*           6           *"
 60 PRINT "    STUD/INST RATIO";TAB(30);"23          17          12"
     :PRINT "    CONTACT HOURS";TAB(31);"*           7           *"
     :PRINT "    INSTRUCTOR ID";TAB(30);"24          18          13"
     :PRINT "    INSTRUCTOR DATA";TAB(31);"*           3           *"
 70 PRINT "    INSTRUCTOR ASSGNMT          26          8           *"
     :PRINT "    BLDG/ROOM NUMBER";TAB(30);"25          19          14"
     :PRINT "    BLDG/ROOM DATA";TAB(31);"*           2           *"
     :RETURN
 80 GOSUB '102(4,13)
     :PRINT "KEY NUMBER INDICATING ACTION TO BE TAKEN."
     :PRINT "(IF NO ADDITIONAL MAINTENANCE REQUIRED, KEY RETURN)    ";
     :J=0
     :INPUT J
     :ON JGOSUB  100 ,2350 ,2040 , 100 , 100 , 590 ,1420 ,2120 , 100
     , 100 ,2370 ,2380 ,2420 ,2390 , 100 ,2430 ,2450 ,2440 ,2460 ,
    2470 , 530 , 100 ,1400 ,1570 ,2330 ,2080 ,2400
100 IF J=0 THEN  110
     :RETURN
110 PRINT HEX(03),"DATA TAPE COMPLETE?(Y/N)"
     :INPUT I$
     :IF I$="Y" THEN  120
     :DATA SAVE END
120 G$(1)="END"
     :DATA SAVE BT (N=100)G$()
     :END
130 DEFFN'60
     :I=0
140 I=I+1
150 KEYIN I$, 170 , 160
     :GOTO  150
160 GOSUB '0
170 IF I<11 THEN  180
```

PROGRAM LISTING

```
      :IF I$=HEX(08) THEN  190
      :IF I$<>HEX(0D) THEN  150
      :RETURN
 120 IF I$=HEX(08) THEN  190
      :IF I$=HEX(0D) THEN  150
      :GOTO  220
 190 IF I>1 THEN  200
      :GOTO  150
 200 PRINT HEX(082D08);
      :IF I=3 THEN  210
      :IF I=7 THEN  210
      :I=I-1
      :GOTO  150
 210 PRINT HEX(082D08);
      :I=I-2
      :GOTO  150
 220 ON IGOTO  230 , 100 , 250 , 260 , 260 , 100 , 260 , 260 , 260 ,
      280
 230 W$="AJKSY"
      :FOR W=1 TO 5
      :IF I$=STR(W$,W,1) THEN  240
      :NEXT W
      :GOTO  150
 240 PRINT I$;"-";
      :STR(K$,I)=I$
      :STR(K$,I+1)="-"
      :I=I+2
      :IF I$<HEX(40) THEN  150
      :STR(K$,3,1)=" "
      :GOTO  150
 250 IF I$<>" " THEN  260
      :PRINT I$;
      :STR(K$,I)=I$
      :GOTO  140
 260 IF I$>HEX(5A) THEN  150
      :IF I$<HEX(30) THEN  150
      :IF I$<HEX(3A) THEN  270
      :IF I$>HEX(40) THEN  270
      :GOTO  150
 270 IF I=5 THEN  240
      :PRINT I$;
      :STR(K$,I)=I$
      :GOTO  140
 280 PRINT I$;
      :STR(K$,I)=I$
      :GOTO  140
 290 DEFFN'105(I9$,C$,E)
      :SELECT PRINT 005(132)
      :@1$=","
      :J=1
      :J9=LEN(I9$)
      :FOR I=1 TO J9
```

PROGRAM LISTING

```
      :PRINT C$;
      :NEXT I
      :FOR I=1 TO J9
      :PRINT HEX(08);
      :NEXT I
      :GOTO  375
310 PRINT Q$;
      :STR(Q1$,J,1)=Q$
      :IF J>=J9 THEN  320
      :J=J+1
      :GOTO  375
320 J=J+1
      :IF E=0 THEN  340
      :GOTO  370
340 KEYIN Q$, 350 , 160
      :GOTO  340
350 IF HEX(0D)<>Q$ THEN  360
      :GOTO  370
360 IF Q$<>HEX(08) THEN  340
      :GOTO  480
370 SELECT PRINT 005(64)
      :RETURN
375 I1$=STR(I9$,J,1)
380 KEYIN Q$, 400 , 160
      :GOTO  380
400 IF Q$<>HEX(08) THEN  410
      :GOTO  480
410 IF Q$<>HEX(0D) THEN  420
      :GOTO  370
420 IF I1$="↑" THEN  310
      :IF I1$="#" THEN  450
440 IF Q$<HEX(41) THEN  375
      :IF Q$>HEX(5A) THEN  375
      :GOTO  310
450 IF Q$<HEX(30) THEN  375
      :IF Q$>HEX(39) THEN  375
      :GOTO  310
480 IF J=1 THEN  380
      :STR(Q1$,J-1,1)=" "
      :J=J-1
      :PRINT HEX(08);C$;HEX(08);
      :GOTO  380
490 DEFFN'102(Q1,Q5)
      :PRINT HEX(01)
      :IF Q5=0 THEN  500
      :FOR I=1 TO Q5
      :PRINT HEX(0A);
      :NEXT I
500 IF Q1=0 THEN  510
      :FOR I=1 TO Q1
      :PRINT HEX(09);
      :NEXT I
```

PROGRAM LISTING

```
 510 RETURN
 515 DEFFN'19
      :PRINT "----------";HEX(08080808080808080808);
      :GOSUB '60
      :RETURN
 520 DEFFN'35(M$,N$)
      :PRINT HEX(03)
      :GOSUB '102(5,3)
      :PRINT "* ";M$;" THE FOLLOWING ";N$
      :RETURN
 530 GOSUB '35("ADD","COURSE")
      :PRINT
      :PRINT "    CDP NUMBER....."
      :PRINT
      :PRINT "    CIN NUMBER....."
      :PRINT
      :PRINT "    CRS SHORT TITLE...."
      :PRINT
      :PRINT "    ACTIVITY/DEPT CODE."
      :PRINT
 540 GOSUB '102(24,5)
      :GOSUB '105("↑↑↑↑","-",0)
      :IF Q1$=" " THEN  540
      :C2$=Q1$
      :STR(T$,7)=Q1$
      :GOSUB '102(24,7)
      :GOSUB '19
      :STR(T$,11)=K$
      :PRINT HEX(0C)
 550 GOSUB '102(24,9)
      :GOSUB '5
      :IF Q1$=" " THEN  550
      :STR(T$,21)=Q1$
 560 GOSUB '102(24,11)
      :GOSUB '105("↑↑↑↑","-",0)
      :IF Q1$=" " THEN  560
      :S1$=Q1$
      :STR(T$,3,4)=Q1$
      :STR(T$,1,2)="21"
      :GOSUB '87
      :GOSUB '222
      :RETURN
 570 DEFFN'0
      :SELECT P9
      :PRINT HEX(03);
      :FOR I=1 TO 15
      :PRINT "ABORT";HEX(0A);
      :NEXT I
      :SELECT P
      :GOTO   40
 580 DEFFN'40(R,N$)
      :PRINT HEX(03)
```

PROGRAM LISTING

```
      :GOSUB '102(4,R)
      :PRINT "*";N$;" DATA FOR THE FOLLOWING COURSE"
      :GOSUB '55(R+2)
      :GOSUB '56(R+4)
      :RETURN
 590 GOSUB '40(5,"CHANGE COURSE")
      :GOSUB '222
      :RETURN
 600 DEFFN'55(N)
      :GOSUB '102(5,N)
      :PRINT "CDP NUMBER ";
      :GOSUB '105("↑##↑","-",0)
      :IF Q1$=" " THEN  600
      :C2$=Q1$
      :RETURN
 610 DEFFN'56(N)
      :PRINT
      :GOSUB '102(5,N)
      :PRINT "ACTIVITY/DEPT CODE "
      :PRINT "      (IF SAME AS PREVIOUS TRANSACTION (";S1$;"),KEY RETU
      RN.)"
 620 GOSUB '102(24,N)
      :GOSUB '105("↑↑↑↑","-",0)
      :IF Q1$=" " THEN  630
      :S1$=Q1$
 630 IF S1$=" " THEN  620
      :RETURN
 640 DEFFN'72(N$)
      :PRINT N$;": CONVEN/LENGTH/CAP/BCAP/PLN INPUT/PLN B-INPUT"
      :PRINT TAB(23);"/      /   /    /          /"
      :RETURN
 650 DEFFN'222
      :PRINT HEX(03)
      :PRINT "            ********** COURSE CDP # ";C2$;" **********"
      :GOSUB '72("01 CURRENT YEAR")
      :GOSUB '72("02 CUR YEAR  +1")
      :GOSUB '72("03 CUR YEAR  +2")
 651 PRINT "04 STUDENT CLASS CAPACITIES:"
      :PRINT TAB(8);"MAX=      MIN=      PERS=      EQUIP=      SPACE="
      :PRINT "05 HISTORICAL CRS RATES:"
      :PRINT "   FAILURE=        NONAC DIS=        NOSHOW=        SETBACK
      ="
 660 GOSUB '80
      :PRINT HEX(03)
      :PRINT "            ********** COURSE CDP # ";C2$;" **********"
      :PRINT "06 ACT/DEPT=      CIN=";TAB(34);"CRS TITLE="
      :PRINT
      :PRINT "07 COURSE UTILIZATION=      PLANNED AOB=      BACKLOG="
 670 PRINT
      :PRINT "08 DATE NEXT COURSE REVIEW=YY/MM/DD    STAFF UIC="
      :PRINT
      :PRINT "09 CCMM=";TAB(20);"PRIORITY CODE="
```

PROGRAM LISTING

```
      :PRINT
      :PRINT "10 COURSE TYPE=     NEC AWARDED="
      :PRINT
      :PRINT "11 OPTIONAL1=          OPTIONAL2=          OPTIONAL3="
      :GOSUB '90
      :RETURN
 680 DEFFN'21
      :GOSUB '102(0,12)
      :PRINT "KEY LINE NUMBER CONTAINING DATA TO BE ENTERED OR CHANGED
      ."
      :PRINT "TO SKIP A DATA ITEM ON A LINE KEY RETURN; CURSOR WILL AD
      VANCE TO"
      :PRINT "THE NEXT DATA ITEM. ";
      :RETURN
 690 DEFFN'80
 700 GOSUB '21
      :PRINT "TO EDIT REMAINING COURSE DATA, RETURN ";
      :Y=6
      :INPUT Y
      :CONVERT Y TO J$,(#)
      :ON YGOSUB  710 , 800 , 820 ,1040 , 880
      :IF Y<6 THEN  700
      :RETURN
 710 GOSUB '102(17,2)
      :GOSUB '81
      :RETURN
 720 DEFFN'101(N$)
      :GOSUB '86("06")
      :STR(T$,11)=N$
      :STR(T$,11+LEN(N$))=J$
      :STR(T$,17)=Q1$
      :GOSUB '87
      :RETURN
 730 DEFFN'81
      :GOSUB '3
      :IF Q1$=" " THEN  740
      :GOSUB '101("CONV")
 740 GOSUB '102(24,Y*2)
      :GOSUB '4
      :IF Q1$=" " THEN  750
      :GOSUB '101("LEN")
 750 GOSUB '102(31,Y*2)
      :GOSUB '3
      :IF Q1$=" " THEN  760
      :GOSUB '101("CAP")
 760 GOSUB '102(35,Y*2)
      :GOSUB '3
      :IF Q1$=" " THEN  770
      :GOSUB '101("BCAP")
 770 GOSUB '102(40,Y*2)
      :GOSUB '105("#####","-",0)
      :IF Q1$=" " THEN  780
```

PROGRAM LISTING

```
      :GOSUB '101("DMD")
 780 GOSUB '102(50,Y*2)
      :GOSUB '105("#####","-",0)
      :IF Q1$=" " THEN  790
      :GOSUB '101("BDMD")
 790 RETURN
 800 GOSUB '102(17,4)
 810 GOSUB '81
      :RETURN
 820 GOSUB '102(17,6)
      :GOTO  810
 830 DEFFN'3
      :GOSUB '105("###","-",0)
      :RETURN
 840 DEFFN'4
      :GOSUB '105("↑↑↑↑","-",0)
      :RETURN
 845 DEFFN'5
      :GOSUB '105("↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑","-",0)
      :RETURN
 850 DEFFN'86(R$)
      :STR(T$,1,2)=R$
      :STR(T$,3,4)=S1$
      :STR(T$,7,4)=C2$
      :RETURN
 860 DEFFN'87
      :G$(1)=T$
      :DATA SAVE BT (N=100)G$()
      :T$,G$(1)=" "
      :RETURN
 870 DEFFN'100(N$)
      :GOSUB '86("06")
      :STR(T$,11)=N$
      :STR(T$,17)=Q1$
      :GOSUB '87
      :PRINT
      :RETURN
 880 GOSUB '102(11,10)
      :GOSUB '85
      :IF LEN(Q1$)=1 THEN  890
      :GOSUB '100("PFAIL")
 890 GOSUB '102(28,10)
      :GOSUB '85
      :IF LEN(Q1$)=1 THEN  900
      :GOSUB '100("NADIS")
 900 GOSUB '102(42,10)
      :GOSUB '85
      :IF LEN(Q1$)=1 THEN  910
      :GOSUB '100("NOSHOW")
 910 GOSUB '102(57,10)
      :GOSUB '85
      :IF LEN(Q1$)=1 THEN  920
```

PROGRAM LISTING

```
      :GOSUB '100("SETB")
 920 RETURN
 930 DEFFN'85
      :STR(Q1$,1)="."
      :J=1
      :PRINT "--.-%";HEX(0808080808);
 940 KEYIN Q$, 950 , 160
      :GOTO  940
 950 IF J<4 THEN  960
      :IF Q$=HEX(08) THEN  960
      :IF Q$<>HEX(0D) THEN  940
      :RETURN
 960 IF Q$<>HEX(08) THEN  970
      :GOTO 1020
 970 IF Q$<>HEX(0D) THEN  980
      :PRINT
      :RETURN
 980 IF Q$<HEX(30) THEN  940
      :IF Q$>HEX(39) THEN  940
      :PRINT Q$;
      :STR(Q1$,J+1,1)=Q$
      :IF J=2 THEN  990
      :J=J+1
      :GOTO  940
 990 PRINT ".";
      :J=J+1
      :GOTO  940
1020 IF J=1 THEN  940
      :IF J=3 THEN 1030
      :STR(Q1$,J,1)=" "
      :J=J-1
      :PRINT HEX(082D08);
      :GOTO  940
1030 STR(Q1$,J,1)=" "
      :J=J-1
      :PRINT HEX(08082D2E0808);
      :GOTO  940
1040 GOSUB '102(12,8)
      :GOSUB '3
      :IF Q1$=" " THEN 1050
      :GOSUB '100("MAXCAP")
1050 GOSUB '102(21,8)
      :GOSUB '3
      :IF Q1$=" " THEN 1060
      :GOSUB '100("MINCAP")
1060 GOSUB '102(31,8)
      :GOSUB '3
      :IF Q1$=" " THEN 1070
      :GOSUB '100("PCAP")
1070 GOSUB '102(42,8)
      :GOSUB '3
      :IF Q1$=" " THEN 1080
```

```
      :GOSUB '100("ECAP")
1080 GOSUB '102(53,8)
      :GOSUB '3
      :IF Q1$=" " THEN 1090
      :GOSUB '100("SCAP")
1090 RETURN
1100 DEFFN'90
1110 GOSUB '21
      :PRINT "IF NO MORE EDITING REQ'D,KEY RETURN. ";
      :Y=0
      :INPUT Y
      :ON Y-5GOTO 1120 ,1130 ,1310 ,1330 ,1350 ,1370
      :IF Y<>0 THEN 1110
      :RETURN
1120 GOSUB '102(12,1)
      :GOSUB '4
      :IF Q1$=" " THEN 1121
      :GOSUB '100("SCH")
1121 GOSUB '102(22,1)
      :GOSUB '19
      :Q1$=K$
      :GOSUB '100("CTLG")
      :GOSUB '102(44,1)
      :GOSUB '5
      :IF Q1$=" " THEN 1110
      :GOSUB '100("CRS")
      :GOTO 1110
1130 GOSUB '102(22,3)
      :PRINT "---%";HEX(08080808);
      :GOSUB '3
      :IF Q1$=" " THEN 1140
      :GOSUB '100("UCAP")
1140 GOSUB '102(40,3)
      :GOSUB '4
      :IF Q1$=" " THEN 1150
      :GOSUB '100("PAOB")
1150 GOSUB '102(54,3)
      :GOSUB '4
      :IF Q1$=" " THEN 1110
      :GOSUB '100("BKLG")
      :GOTO 1110
1160 DEFFN'20
      :I=1
      :Q1$=" "
1170 KEYIN Q$,1180 , 160
      :GOTO 1170
1180 IF I<7 THEN 1190
      :IF Q$=HEX(08) THEN 1190
      :IF Q$<>HEX(0D) THEN 1170
      :RETURN
1190 IF Q$<>HEX(08) THEN 1200
      :GOTO 1290
```

```
1200 IF Q$<>HEX(0D) THEN 1210
      :RETURN
1210 IF I=1 THEN 1220
      :IF I=3 THEN 1240
      :IF I=5 THEN 1250
      :IF Q$<HEX(30) THEN 1170
      :IF Q$>HEX(39) THEN 1170
      :IF I=6 THEN 1230
      :GOSUB '91
      :PRINT "/";
      :GOTO 1170
1220 IF Q$<HEX(37) THEN 1170
      :IF Q$>HEX(39) THEN 1170
1230 GOSUB '91
      :GOTO 1170
1240 IF Q$<HEX(30) THEN 1170
      :IF Q$>HEX(31) THEN 1170
      :GOTO 1230
1250 IF Q$<HEX(30) THEN 1170
      :IF Q$>HEX(33) THEN 1170
      :GOTO 1230
1260 DEFFN'91
      :PRINT Q$;
      :STR(Q1$,I,1)=Q$
      :I=I+1
      :RETURN
1290 IF I=1 THEN 1170
      :IF I=3 THEN 1300
      :IF I=5 THEN 1300
      :STR(Q1$,I-1,1)=" "
      :I=I-1
      :PRINT HEX(082008);
      :GOTO 1170
1300 STR(Q1$,I-1,1)=" "
      :I=I-1
      :PRINT HEX(0808202F0808);
      :GOTO 1170
1310 GOSUB '102(27,5)
      :GOSUB '20
      :PRINT HEX(0C)
      :IF Q1$=" " THEN 1320
      :GOSUB '100("RDTE")
1320 GOSUB '102(49,5)
      :GOSUB '105("######","-",0)
      :IF Q1$=" " THEN 1110
      :GOSUB '100("SUIC")
      :GOTO 1110
1330 GOSUB '102(8,7)
      :GOSUB '105("↑↑↑↑↑↑↑↑↑↑","-",0)
      :IF Q1$=" " THEN 1340
      :GOSUB '100("CCMM")
1340 GOSUB '102(34,7)
```

PROGRAM LISTING

```
      :GOSUB '105("##","-",0)
      :IF Q1$=" " THEN 1110
      :GOSUB '100("PRTY")
      :GOTO 1110
1350 GOSUB '102(15,9)
      :GOSUB '105("↑#","-",0)
      :IF Q1$=" " THEN 1360
      :GOSUB '100("CTYPE")
1360 GOSUB '102(31,9)
      :GOSUB '4
      :IF Q1$=" " THEN 1110
      :GOSUB '100("PNEC")
      :GOTO 1110
1370 GOSUB '102(13,11)
      :GOSUB '105("↑↑↑↑↑↑","-",0)
      :IF Q1$=" " THEN 1380
      :GOSUB '100("OPT1")
1380 GOSUB '102(31,11)
      :GOSUB '105("↑↑↑↑↑↑","-",0)
      :IF Q1$=" " THEN 1390
      :GOSUB '100("OPT2")
1390 GOSUB '102(49,11)
      :GOSUB '105("↑↑↑↑↑↑","-",0)
      :IF Q1$=" " THEN 1110
      :GOSUB '100("OPT3")
      :GOTO 1110
1400 GOSUB '40(5,"ADD STUDENT/INSTRUCTOR RATIO")
1410 GOSUB '41
      :GOSUB '86("23")
      :GOSUB '44(3)
      :STR(T$,11)=N1$
      :GOSUB '42(0)
      :GOSUB '87
      :GOSUB '33("RATIO DATA","COURSE")
      :IF Y=1 THEN 1410
      :RETURN
1420 GOSUB '40(5,"CHANGE CONTACT HOUR")
1430 GOSUB '41
      :GOSUB '44(3)
      :GOSUB '42(1)
      :GOSUB '33("CONTACT HOUR CHANGES","COURSE")
      :IF Y=1 THEN 1430
      :RETURN
1440 DEFFN'41
      :PRINT HEX(03)
      :GOSUB '102(4,1)
      :PRINT "CDP#";C2$
      :RETURN
1450 DEFFN'44(N)
      :GOSUB '102(7,N)
      :PRINT "STUDENT/INSTRUCTOR RATIO = ---/1";HEX(0808080808);
      :GOSUB '3
```

PROGRAM LISTING

```
      :IF Q1$=" " THEN 1450
      :N1$=Q1$
      :RETURN
1460 DEFFN'42(N)
1470 GOSUB '102(7,5)
      :PRINT "CONTACT HOURS AT THIS RATIO = ";
      :GOSUB '4
      :IF Q1$=" " THEN 1480
      :GOTO 1490
1480 IF N=0 THEN 1470
      :GOTO 1510
1490 IF N<>0 THEN 1500
      :STR(T$,14)=Q1$
      :GOTO 1510
1500 GOSUB '43("F2")
1510 GOSUB '102(7,7)
      :PRINT "RATIO TYPE ="
      :PRINT TAB(9);"1=LECTURE"
      :PRINT TAB(9);"2=LAB"
      :PRINT TAB(9);"3=COMB LEC/LAB"
      :GOSUB '102(20,7)
      :GOSUB '105("#","-",0)
      :IF Q1$=" " THEN 1520
      :GOTO 1530
1520 IF N=0 THEN 1510
      :RETURN
1530 IF N<>0 THEN 1540
      :STR(T$,18)=Q1$
      :RETURN
1540 GOSUB '43("F3")
      :RETURN
1550 DEFFN'43(N$)
      :GOSUB '86("07")
      :STR(T$,11)=N1$
      :STR(T$,14)=N$
      :STR(T$,20)=Q1$
      :GOSUB '87
      :RETURN
1560 DEFFN'31(N)
      :GOSUB '102(5,N)
      :PRINT "INSTRUCTOR ID(SSN) =----------";
      :FOR J=1 TO 11
      :PRINT HEX(08);
      :NEXT J
      :GOSUB '59
      :IF Q1$=" " THEN 1560
      :IO$=Q1$
      :PRINT
      :RETURN
1570 GOSUB '35("ADD","INSTRUCTOR")
      :N1$=" "
      :GOSUB '31(5)
```

```
      :GOSUB '102(5,7)
      :PRINT "LAST NAME"
1580 GOSUB '102(16,7)
      :GOSUB '105("↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑","-",0)
      :IF Q1$=" " THEN 1580
      :STR(N1$,3)=Q1$
      :PRINT
      :PRINT "     FIRST,MIDDLE INITIALS"
1590 GOSUB '102(27,8)
      :GOSUB '105("↑↑","-",0)
      :IF Q1$=" " THEN 1590
      :STR(N1$,1,2)=Q1$
1600 GOSUB '102(5,10)
      :PRINT "RATE,RANK =";
      :GOSUB '105("↑↑↑↑↑","-",0)
      :IF Q1$=" " THEN 1600
      :STR(T$,16,5)=Q1$
      :GOSUB '102(5,12)
      :PRINT "ASSIGNED TO"
      :PRINT
      :GOSUB '56(13)
      :GOSUB '86("24")
      :STR(T$,7,9)=IO$
      :STR(T$,21)=N1$
      :GOSUB '87
      :GOSUB '124
      :RETURN
1610 DEFFN'59
      :I=1
      :Q1$=" "
1620 KEYIN Q$,1630 , 160
      :GOTO 1620
1630 IF I<10 THEN 1640
      :IF Q$=HEX(08) THEN 1640
      :IF Q$<>HEX(0D) THEN 1620
      :RETURN
1640 IF Q$<>HEX(08) THEN 1650
      :GOTO 1690
1650 IF Q$<HEX(30) THEN 1620
      :IF Q$>HEX(39) THEN 1620
      :IF I=3 THEN 1660
      :IF I=5 THEN 1660
      :GOSUB '91
      :GOTO 1620
1660 GOSUB '91
      :PRINT "-";
      :GOTO 1620
1690 IF I=1 THEN 1620
      :IF I=4 THEN 1700
      :IF I=6 THEN 1700
      :STR(Q1$,I-1)=" "
      :I=I-1
```

```
      :PRINT HEX(082D08);
      :GOTO 1620
1700 STR(Q1$,I-1)=" "
      :I=I-1
      :PRINT HEX(08082D08);
      :GOTO 1620
1710 DEFFN'88(N$)
      :GOSUB '86("03")
      :STR(T$,7)=IO$
      :STR(T$,16)=N$
      :STR(T$,22)=Q1$
      :GOSUB '87
      :RETURN
1720 DEFFN'92
      :PRINT HEX(03)
      :PRINT "     ID = ";STR(IO$,1,3);"-";STR(IO$,4,2);"-";STR(IO$,6,4
      );"     NAME = ";STR(N1$,3,16)
      :RETURN
1730 DEFFN'124
      :GOSUB '92
      :PRINT
      :PRINT "01 NAME";TAB(27);"INITIALS"
      :PRINT
      :PRINT "02 RATE =";TAB(27);"ACT/DEPT ="
      :PRINT
1735 PRINT "03 REPORT DATE = YY/MM/DD    ROTATE DATE = YY/MM/DD"
      :PRINT
      :PRINT "04 PRIMARY NEC =        SECONDARY NEC ="
      :PRINT
      :PRINT "05 ADDITIONAL NECS ="
1740 GOSUB '21
      :PRINT "TO EDIT REMAINING INST DATA, RETURN ";
      :Y=6
      :INPUT Y
      :ON YGOTO 1750 ,1770 ,1780 ,1800 ,1820 ,1850
      :GOTO 1740
1750 GOSUB '102(8,2)
      :GOSUB '5
      :IF Q1$=" " THEN 1760
      :GOSUB '88("NME")
1760 GOSUB '102(36,2)
      :GOSUB '105("↑↑","-",0)
      :IF Q1$=" " THEN 1740
      :GOSUB '88("IN")
      :GOTO 1740
1770 GOSUB '102(10,4)
      :GOSUB '105("↑↑↑↑↑","-",0)
      :IF Q1$=" " THEN 1775
      :GOSUB '88("RATE")
      :GOTO 1775
1775 GOSUB '102(38,4)
      :GOSUB '4
```

PROGRAM LISTING

```
      :IF Q1$=" " THEN 1740
      :GOSUB '88("SCH")
      :GOTO 1740
1780 GOSUB '102(17,6)
      :GOSUB '20
      :IF Q1$=" " THEN 1790
      :Q1$=STR(Q1$,1,4)
      :GOSUB '88("REP")
      :PRINT HEX(0C)
      :GOTO 1790
1790 GOSUB '102(43,6)
      :GOSUB '20
      :IF Q1$=" " THEN 1740
      :Q1$=STR(Q1$,1,4)
      :GOSUB '88("ROT")
      :PRINT HEX(0C)
      :GOTO 1740
1800 GOSUB '102(17,8)
      :GOSUB '4
      :IF Q1$=" " THEN 1810
      :GOSUB '88("PNEC")
1810 GOSUB '102(39,8)
      :GOSUB '4
      :IF Q1$=" " THEN 1740
      :GOSUB '88("SNEC")
      :GOTO 1740
1820 GOSUB '102(21,10)
      :GOSUB '4
      :IF Q1$=" " THEN 1830
      :GOSUB '88("NEC3")
1830 GOSUB '102(28,10)
      :GOSUB '4
      :IF Q1$=" " THEN 1840
      :GOSUB '88("NEC4")
1840 GOSUB '102(35,10)
      :GOSUB '4
      :IF Q1$=" " THEN 1740
      :GOSUB '88("NEC5")
      :GOTO 1740
1850 GOSUB '92
      :PRINT
      :PRINT "  **NOTE: ALL INSTRUCTOR ACTIVITY DATA ARE HOURS/YEAR"
      :PRINT
      :PRINT "06 STUDENT CONTACT = ----        CURRICULUM DEVELOPMENT =
       ----"
      :PRINT "07 CLASS PREP/RELATED DUT=----  INST TRNG(BREAK-IN) = --
      --"
1860 PRINT "08 EQUIP-TRNG/MAINT = ----       CROSS TRNG(LOW STUD LOAD)=
      ----"
      :PRINT "09 FORMAL INST SCH = ----        FACTORY TRNG/WKSHOPS = -
      ---"
      :PRINT "10 SPEC FLEET ASSISTANCE= ----  MIL DUT/PHYS TRNG/OTHER
```

```
      = ----"
1870 PRINT "11 SUPERVISION/ADMIN = ----     ANNUAL LEAVE = ----"
      :PRINT "12 SICK LEAVE = ----          SPECIAL LIBERTY = ----"
1880 GOSUB '21
      :PRINT "IF NO MORE EDITING REQUIRED,KEY RETURN.";
      :Y=0
      :INPUT Y
      :ON Y-5GOTO 1890 ,1910 ,1930 ,1950 ,1970 ,1990 ,2010
      :IF Y>0 THEN 1880
      :RETURN
1890 GOSUB '102(21,4)
      :GOSUB '4
      :IF Q1$=" " THEN 1900
      :GOSUB '88("CONT")
1900 GOSUB '102(57,4)
      :GOSUB '4
      :IF Q1$=" " THEN 1880
      :GOSUB '88("CDEV")
      :GOTO 1880
1910 GOSUB '102(26,5)
      :GOSUB '4
      :IF Q1$=" " THEN 1920
      :GOSUB '88("PREP")
1920 GOSUB '102(54,5)
      :GOSUB '4
      :IF Q1$=" " THEN 1880
      :GOSUB '88("BKIN")
      :GOTO 1880
1930 GOSUB '102(22,6)
      :GOSUB '4
      :IF Q1$=" " THEN 1940
      :GOSUB '88("EQMT")
1940 GOSUB '102(58,6)
      :GOSUB '4
      :IF Q1$=" " THEN 1880
      :GOSUB '88("CTRN")
      :GOTO 1880
1950 GOSUB '102(21,7)
      :GOSUB '4
      :IF Q1$=" " THEN 1960
      :GOSUB '88("ISCH")
1960 GOSUB '102(55,7)
      :GOSUB '4
      :IF Q1$=" " THEN 1880
      :GOSUB '88("FTRN")
      :GOTO 1880
1970 GOSUB '102(26,8)
      :GOSUB '4
      :IF Q1$=" " THEN 1980
      :GOSUB '88("FLET")
1980 GOSUB '102(58,8)
      :GOSUB '4
```

```
      :IF Q1$=" " THEN 1880
      :GOSUB '88("MDUT")
      :GOTO 1880
1990 GOSUB '102(23,9)
      :GOSUB '4
      :IF Q1$=" " THEN 2000
      :GOSUB '88("ADMN")
2000 GOSUB '102(47,9)
      :GOSUB '4
      :IF Q1$=" " THEN 1880
      :GOSUB '88("ANLV")
      :GOTO 1880
2010 GOSUB '102(16,10)
      :GOSUB '4
      :IF Q1$=" " THEN 2020
      :GOSUB '88("SKLV")
2020 GOSUB '102(50,10)
      :GOSUB '4
      :IF Q1$=" " THEN 1880
      :GOSUB '88("SLIB")
      :GOTO 1880
2030 DEFFN'30(N,N$)
      :PRINT HEX(03)
      :GOSUB '102(5,N)
      :PRINT N$;" DATA FOR THE FOLLOWING INSTRUCTOR:"
      :RETURN
2040 GOSUB '30(4,"CHANGE")
      :GOSUB '31(6)
      :GOSUB '56(8)
      :N1$=" "
      :GOSUB '124
      :RETURN
2050 DEFFN'32(N)
      :GOSUB '102(5,N)
      :PRINT "ASSIGN FLAG ="
      :PRINT "      (ENTER 0 IF NOT TEACHING ABOVE COURSE; ELSE ENTER 1
      )"
      :GOSUB '102(19,N)
      :GOSUB '105("#","-",0)
      :RETURN
2060 DEFFN'33(N$,M$)
      :GOSUB '102(5,13)
      :PRINT "TO ENTER ADDITIONAL ";N$;" FOR THE ABOVE"
      :PRINT TAB(5);M$;" TYPE 1, ELSE KEY RETURN";
      :Y=0
      :INPUT Y
      :RETURN
2070 DEFFN'34(N$)
      :GOSUB '86("08")
      :STR(T$,11)=IO$
      :STR(T$,20)=N$
      :STR(T$,26)=Q1$
```

PROGRAM LISTING

```
      :GOSUB '87
      :RETURN
2080 GOSUB '30(0,"ADD INSTR ASSIGNMENT")
      :GOSUB '31(2)
2090 PRINT HEX(0C)
      :GOSUB '102(38,2)
      :PRINT "IS ---% QUALIFIED"
      :PRINT "     TO INSTRUCT IN THE FOLLOWING COURSE:"
2100 GOSUB '102(41,2)
      :GOSUB '3
      :IF Q1$=" " THEN 2100
      :STR(T$,20)=Q1$
      :GOSUB '55(5)
      :GOSUB '56(7)
2110 GOSUB '32(10)
      :IF Q1$=" " THEN 2110
      :STR(T$,24)=Q1$
      :GOSUB '86("26")
      :STR(T$,11,9)=IO$
      :GOSUB '87
      :GOSUB '33("COURSE ASSIGNMENTS","INSTRUCTOR")
      :IF Y=1 THEN 2090
      :RETURN
2120 GOSUB '30(0,"CHANGE ASSIGNMENT")
      :GOSUB '31(2)
      :PRINT HEX(0C)
2130 GOSUB '55(3)
      :GOSUB '56(4)
      :GOSUB '102(5,7)
      :PRINT "NEW ASSIGNMENT DATA:"
      :PRINT "      % QUALIFIED = ---%";HEX(08080808);
      :GOSUB '3
      :IF Q1$=" " THEN 2140
      :GOSUB '34("F2")
2140 GOSUB '32(10)
      :IF Q1$=" " THEN 2150
      :GOSUB '34("F3")
2150 GOSUB '33("CHANGES","INSTRUCTOR")
      :IF Y=1 THEN 2130
      :RETURN
2160 DEFFN'36(N)
      :GOSUB '102(5,N)
      :PRINT "BUILDING/ROOM NUMBER = ----/----";HEX(0808080808080808
      );
      :GOSUB '4
      :IF Q1$=" " THEN 2160
      :STR(T$,11)=Q1$
      :N1$=Q1$
2170 GOSUB '102(33,N)
      :GOSUB '4
      :IF Q1$=" " THEN 2170
      :STR(T$,15)=Q1$
```

PROGRAM LISTING

```
      :STR(N1$,5)=Q1$
      :RETURN
2180 DEFFN'37
      :GOSUB '102(5,9)
      :PRINT "ROOM CAPACITY(#STDT)=---   ROOM TYPE = -"
      :PRINT TAB(30);"(1=LEC,2=LAB,3=BOTH,4=TRNER)"
      :PRINT
      :PRINT "     HOURS REQ'D PER ";C2$;" CONV =----  TOT HR AVAIL/YE
      AR=-----"
      :RETURN
2190 DEFFN'38(R)
2200 GOSUB '102(26,9)
      :GOSUB '3
      :IF Q1$<>" " THEN 2210
      :IF R=0 THEN 2200
      :GOTO 2230
2210 IF R<>0 THEN 2220
      :STR(T$,19)=Q1$
      :GOTO 2230
2220 GOSUB '39("RMCAP")
2230 GOSUB '102(44,9)
      :GOSUB '105("#","-",0)
      :IF Q1$<>" " THEN 2240
      :IF R=0 THEN 2230
      :GOTO 2260
2240 IF R<>0 THEN 2250
      :STR(T$,22)=Q1$
      :GOTO 2260
2250 GOSUB '39("RMTP")
2260 GOSUB '102(32,12)
      :GOSUB '4
      :IF Q1$<>" " THEN 2270
```

```
      :IF R=0 THEN 2260
      :GOTO 2290
2270 IF R<>0 THEN 2280
      :STR(T$,23)=Q1$
      :GOTO 2290
2280 GOSUB '39("REQHRS")
2290 GOSUB '102(56,12)
      :GOSUB '105("#####","-",0)
      :IF Q1$<>" " THEN 2300
      :IF R=0 THEN 2290
      :RETURN
2300 IF R<>0 THEN 2310
      :STR(T$,30)=Q1$
      :RETURN
2310 GOSUB '39("AVHRS")
      :RETURN
2320 DEFFN'39(N$)
      :GOSUB '86("02")
      :STR(T$,11)=N1$
      :STR(T$,19)=N$
      :STR(T$,25)=Q1$
      :GOSUB '87
      :RETURN
2330 GOSUB '40(0,"ADD FACILITY")
2340 GOSUB '86("25")
      :GOSUB '36(7)
      :GOSUB '37
      :GOSUB '38(0)
      :GOSUB '87
      :GOSUB '33("FACILITY DATA","COURSE")
      :IF Y=1 THEN 2340
      :RETURN
2350 GOSUB '40(0,"CHANGE FACILITY")
2360 GOSUB '36(7)
      :PRINT
      :PRINT "      REVISED DATA:"
      :GOSUB '37
      :GOSUB '38(1)
      :GOSUB '33("FACILITY CHANGES","COURSE")
      :IF Y=1 THEN 2360
      :RETURN
2370 GOSUB '40(5,"DELETE ALL")
      :GOSUB '86("11")
      :GOSUB '87
      :RETURN
2380 GOSUB '40(5,"DELETE STUDENT/INSTR RATIO")
      :GOSUB '44(12)
      :GOSUB '86("12")
      :STR(T$,11)=N1$
      :GOSUB '87
      :RETURN
2390 GOSUB '40(5,"DELETE FACILITY")
```

PROGRAM LISTING

```
      :PRINT HEX(03)
      :GOSUB '86("14")
      :GOSUB '36(7)
      :GOSUB '87
      :RETURN
2400 GOSUB '40(2,"DELETE INSTRUCTOR ASSIGNMENT")
2410 GOSUB '31(9)
      :GOSUB '86("27")
      :STR(T$,11)=IO$
      :GOSUB '87
      :GOSUB '33("DELETIONS","COURSE")
      :IF Y=1 THEN 2410
      :RETURN
2420 GOSUB '35("DELETE","INSTRUCTOR")
      :GOSUB '31(7)
      :GOSUB '56(10)
      :GOSUB '86("13")
      :STR(T$,7)=IO$
      :GOSUB '87
      :RETURN
2430 GOSUB '35("CHANGE","CDP NUMBER")
      :GOSUB '102(1,6)
      :PRINT "OLD"
      :PRINT
      :GOSUB '55(6)
      :STR(T$,1)="16"
      :STR(T$,7)=C2$
      :GOSUB '102(1,8)
      :PRINT "NEW"
      :PRINT
      :GOSUB '55(8)
      :GOSUB '56(10)
      :STR(T$,11)=C2$
      :STR(T$,3,4)=S1$
      :GOSUB '87
      :RETURN
2440 GOSUB '35("CHANGE","INSTRUCTOR ID(SSN)")
      :GOSUB '102(1,7)
      :PRINT "OLD"
      :PRINT
      :GOSUB '31(7)
      :STR(T$,7)=IO$
      :GOSUB '102(1,9)
      :PRINT "NEW"
      :PRINT
      :GOSUB '31(9)
      :STR(T$,16)=IO$
      :GOSUB '56(11)
      :STR(T$,1,2)="18"
      :STR(T$,3,4)=S1$
      :GOSUB '87
      :RETURN
2450 GOSUB '40(3,"CHANGE STUDENT/INSTR RATIO")
```

```
       :GOSUB '86("17")
       :GOSUB '102(3,10)
       :PRINT "OLD"
       :PRINT
       :GOSUB '44(10)
       :S$=Q1$
       :GOSUB '102(3,12)
       :PRINT "NEW"
       :PRINT
       :GOSUB '44(12)
       :STR(T$,11)=S$
       :STR(T$,14)=Q1$
       :GOSUB '87
       :RETURN
2460 GOSUB '40(3,"CHANGE BUILDING/ROOM NUMBER")
       :GOSUB '86("19")
       :GOSUB '102(1,10)
       :PRINT "OLD"
       :PRINT
       :GOSUB '36(10)
       :S$=STR(T$,11,8)
       :GOSUB '102(1,12)
       :PRINT "NEW"
       :PRINT
       :GOSUB '36(12)
       :STR(T$,19)=STR(T$,11,8)
       :STR(T$,11,8)=S$
       :GOSUB '87
       :RETURN
2470 GOSUB '35("CHANGE","ACTIVITY/DEPT CODE")
       :GOSUB '102(1,6)
       :PRINT "OLD"
       :PRINT
       :GOSUB '56(6)
       :STR(T$,1)="20"
       :STR(T$,3)=S1$
       :STR(T$,7)=S1$
       :GOSUB '102(1,10)
       :PRINT "NEW"
       :PRINT
       :GOSUB '56(10)
       :STR(T$,11)=S1$
       :GOSUB '87
       :RETURN
```

10SEPT76 PAGE 1
LINE NUMBER CROSS REFERENCE - DBPROMPT

0040 - 0010 0040 0570

0090 - 0040

0100 - 0080 0080 0080 0080 0080 0080 0080 0220 0220

0110 - 0100

0120 - 0110

0140 - 0250 0270 0280

0150 - 0150 0170 0180 0190 0200 0210 0230 0240 0240 0260 0260 0260

0160 - 0150 0340 0380 0940 1170 1620

0170 - 0150

0180 - 0170

0190 - 0170 0180

0200 - 0190

0210 - 0200 0200

0220 - 0180

0230 - 0220

0240 - 0230 0270

0250 - 0220

0260 - 0220 0220 0220 0220 0220 0250

0270 - 0260 0260

0280 - 0220

0310 - 0420 0440 0450

0330 - 0310

0340 - 0330 0340 0360

0350 - 0340

0360 - 0350

0370 - 0330 0350 0410

LINE NUMBER CROSS REFERENCE

0375 - 0290 0310 0440 0440 0450 0450

0380 - 0320 0480 0480

0400 - 0380

0410 - 0400

0420 - 0410

0450 - 0420

0480 - 0360 0400

0500 - 0490

0510 - 0500

0530 - 0080

0540 - 0540

0550 - 0550

0560 - 0560

0590 - 0080

0600 - 0600

0620 - 0630

0630 - 0620

0700 - 0700

0710 - 0700

0740 - 0730

0750 - 0740

0760 - 0750

0770 - 0760

0780 - 0770

0790 - 0780

0800 - 0700

LINE NUMBER CROSS REFERENCE

0810 - 0820

0820 - 0700

0880 - 0700

0890 - 0880

0900 - 0890

0910 - 0900

0920 - 0910

0940 - 0940 0950 0980 0980 0980 0990 1020 1020 1030

0950 - 0940

0960 - 0950 0950

0970 - 0960

0980 - 0970

0990 - 0980

1020 - 0960

1030 - 1020

1040 - 0700

1050 - 1040

1060 - 1050

1070 - 1060

1080 - 1070

1090 - 1080

1110 - 1110 1121 1121 1150 1150 1320 1320 1340 1340 1360 1360 1390 1390

1120 - 1110

1121 - 1120

1130 - 1110

1140 - 1130

LINE NUMBER CROSS REFERENCE

1150 - 1140

1170 - 1170 1180 1210 1210 1210 1220 1220 1230 1240 1240 1250 1250
1290 1290 1300

1180 - 1170

1190 - 1180 1180

1200 - 1190

1210 - 1200

1220 - 1210

1230 - 1210 1240 1250

1240 - 1210

1250 - 1210

1290 - 1190

1300 - 1290 1290

1310 - 1110

1320 - 1310

1330 - 1110

1340 - 1330

1350 - 1110

1360 - 1350

1370 - 1110

1380 - 1370

1390 - 1380

1400 - 0080

1410 - 1410

1420 - 0080

1430 - 1430

1450 - 1450

LINE NUMBER CROSS REFERENCE

1470 - 1480

1480 - 1470

1490 - 1470

1500 - 1490

1510 - 1480 1490 1520

1520 - 1510

1530 - 1510

1540 - 1530

1560 - 1560

1570 - 0080

1580 - 1580

1590 - 1590

1600 - 1600

1620 - 1620 1630 1650 1650 1650 1660 1690 1690 1700

1630 - 1620

1640 - 1630 1630

1650 - 1640

1660 - 1650 1650

1690 - 1640

1700 - 1690 1690

1740 - 1740 1760 1760 1775 1775 1790 1790 1810 1810 1840 1840

1750 - 1740

1760 - 1750

1770 - 1740

1775 - 1770 1770

1780 - 1740

LINE NUMBER CROSS REFERENCE

1790 - 1780 1780

1800 - 1740

1810 - 1800

1820 - 1740

1830 - 1820

1840 - 1830

1850 - 1740

1880 - 1880 1900 1900 1920 1920 1940 1940 1960 1960 1980 1980 2000
2000 2020 2020

1890 - 1880

1900 - 1890

1910 - 1880

1920 - 1910

1930 - 1880

1940 - 1930

1950 - 1880

1960 - 1950

1970 - 1880

1980 - 1970

1990 - 1880

2000 - 1990

2010 - 1880

2020 - 2010

2040 - 0080

2080 - 0080

2090 - 2110

2100 - 2100

LINE NUMBER CROSS REFERENCE

2110 - 2110

2120 - 0080

2130 - 2150

2140 - 2130

2150 - 2140

2160 - 2160

2170 - 2170

2200 - 2200

2210 - 2200

2220 - 2210

2230 - 2200 2210 2230

2240 - 2230

2250 - 2240

2260 - 2230 2240

2330 - 0080

2350 - 0080

2370 - 0080

2380 - 0080

2390 - 0080

2400 - 0080

2420 - 0080

2430 - 0080

2440 - 0080

2450 - 0080

2460 - 0080

2470 - 0080

LINE NUMBER CROSS REFERENCE

2260 - 2260

2270 - 2260

2280 - 2270

2290 - 2260 2270 2290

2300 - 2290

2310 - 2300

2340 - 2340

2360 - 2360

2410 - 2410

VARIABLE CROSS REFERENCE - **DBPROMPT**

C$ - 0230 0290 0480

C2% - 0010 0540 0600 0610 0660 0850 1440 2180

F - 0290 0330

G$() - 0010 0120 0120 0860 0860 0860

I - 0130 0140 0140 0170 0190 0200 0200 0200 0200 0210 0210 0220
0240 0240 0240 0240 0250 0270 0270 0280 0290 0290 0290 0290
0490 0490 0500 0500 0570 0570 1160 1180 1210 1210 1210 1210
1260 1260 1260 1290 1290 1290 1290 1290 1290 1300 1300 1300
1610 1630 1650 1650 1690 1690 1690 1690 1690 1690 1700 1700
1700

I$ - 0010 0010 0010 0110 0110 0150 0170 0170 0180 0180 0230 0240
0240 0240 0250 0250 0250 0260 0260 0260 0260 0270 0270 0280
0280

IO$ - 0010 1560 1600 1710 1720 1720 1720 2070 2110

II$ - 0010 0375 0420 0420

I9$ - 0010 0290 0290 0375

J - 0080 0080 0080 0100 0290 0310 0310 0310 0310 0330 0330 0375
0480 0480 0480 0480 0930 0950 0980 0980 0980 0980 0990 0990
1020 1020 1020 1020 1020 1030 1030 1030 1560 1560

J$ - 0700 0720

J9 - 0290 0290 0290 0310

K$ - 0240 0240 0240 0250 0270 0280 0540 1121

M$ - 0520 0520 2060 2060

N - 0120 0600 0600 0610 0610 0620 0860 1450 1450 1460 1480 1490
1520 1530 1560 1560 2030 2030 2050 2050 2050 2160 2160 2170

N$ - 0010 0520 0520 0580 0580 0640 0640 0720 0720 0720 0870 0870
1550 1550 1710 1710 2030 2030 2060 2060 2070 2070

N1$ - 0010 1410 1450 1550 1570 1580 1590 1600 1720 2040 2160 2170

O - 0010

Q$ - 0310 0310 0340 0350 0360 0380 0400 0410 0440 0440 0450 0450
0940 0950 0950 0960 0970 0980 0980 0980 0980 1170 1180 1180
1190 1200 1210 1210 1220 1220 1240 1240 1250 1250 1260 1260

VARIABLE CROSS REFERENCE

```
          1620 1630 1630 1640 1650 1650

Q1      - 0490 0500 0500

Q1$     - 0010 0290 0310 0480 0540 0540 0540 0550 0550 0560 0560 0560
          0600 0600 0620 0620 0720 0730 0740 0750 0760 0770 0780 0870
          0880 0890 0900 0910 0930 0980 1020 1030 1040 1050 1060 1070
          1080 1120 1121 1121 1130 1140 1150 1160 1260 1290 1300 1310
          1320 1330 1340 1350 1360 1370 1380 1390 1450 1450 1470 1490
          1510 1530 1550 1560 1560 1580 1580 1590 1590 1600 1600 1610
          1690 1700 1710 1750 1760 1770 1775 1780 1780 1780 1790 1790
          1790 1800 1810 1820 1830 1840 1890 1900 1910 1920 1930 1940
          1950 1960 1970 1980 1990 2000 2010 2020 2070 2100 2100 2110
          2110 2130 2140 2160 2160 2160 2170 2170 2170 2200 2210 2230
          2240 2260

Q5      - 0490 0490 0490

R       - 0580 0580 0580 0580 2190 2200 2210 2230 2240

R$      - 0850 0850

S1$     - 0010 0560 0610 0620 0630 0850

T$      - 0010 0540 0540 0550 0560 0560 0720 0720 0720 0850 0850 0850
          0860 0860 0870 0870 1410 1490 1530 1550 1550 1550 1600 1600
          1600 1710 1710 1710 2070 2070 2070 2100 2110 2110 2160 2170
          2210 2240

W       - 0230 0230 0230

W$      - 0230 0230

Y       - 0700 0700 0700 0700 0700 0740 0750 0760 0770 0780 1110 1110
          1110 1110 1410 1430 1740 1740 1740 1880 1880 1880 1880 2060
          2060 2110 2150
```

VARIABLE CROSS REFERENCE

C2$ - 2430 2430

IO$ - 2410 2420 2440 2440

N$ - 2320 2320

N1$ - 2320 2380

Q1$ - 2270 2290 2300 2320 2450 2450

R - 2260 2270 2290 2300

S$ - 2450 2450 2460 2460

S1$ - 2430 2440 2470 2470 2470

T$ - 2270 2300 2320 2320 2320 2380 2410 2420 2430 2430 2430 2430
2440 2440 2440 2440 2450 2450 2460 2460 2460 2460 2470 2470
2470 2470

Y - 2340 2360 2410

SPECIAL FUNCTION CROSS REFERENCE - DBPROMPT

```
' 0 -   0160 0570

' 3 -   0730 0750 0760 0830 1040 1050 1060 1070 1080 1130 1450 2100
       2130 2200

' 4 -   0740 0840 1120 1140 1150 1360 1470 1775 1800 1810 1820 1830
       1840 1890 1900 1910 1920 1930 1940 1950 1960 1970 1980 1990
       2000 2010 2020 2160 2170 2260

' 5 -   0550 0845 1121 1750

' 19 -  0515 0540 1121

' 20 -  1160 1310 1780 1790

' 21 -  0680 0700 1110 1740 1880

' 30 -  2030 2040 2080 2120

' 31 -  1560 1570 2040 2080 2120

' 32 -  2050 2110 2140

' 33 -  1410 1430 2060 2110 2150

' 34 -  2070 2130 2140

' 35 -  0520 0530 1570

' 36 -  2160

' 37 -  2180

' 38 -  2190

' 39 -  2220 2250

' 40 -  0580 0590 1400 1420

' 41 -  1410 1430 1440

' 42 -  1410 1430 1460

' 43 -  1500 1540 1550

' 44 -  1410 1430 1450

' 50 -  0040 0050

' 55 -  0580 0600 2100 2130

' 56 -  0580 0610 1600 2040 2100 2130
```

SPECIAL FUNCTION CROSS REFERENCE

' 59 - 1560 1610

' 60 - 0130 0515

' 72 - 0640 0650 0650 0650

' 80 - 0660 0690

' 81 - 0710 0730 0810

' 85 - 0880 0890 0900 0910 0930

' 86 - 0720 0850 0870 1410 1550 1600 1710 2070 2110

' 87 - 0560 0720 0860 0870 1410 1550 1600 1710 2070 2110

' 88 - 1710 1750 1760 1770 1775 1780 1790 1800 1810 1820 1830 1840
1890 1900 1910 1920 1930 1940 1950 1960 1970 1980 1990 2000
2010 2020

' 90 - 0670 1100

' 91 - 1210 1230 1260 1650 1660

' 92 - 1720 1730 1850

' 100 - 0870 0880 0890 0900 0910 1040 1050 1060 1070 1080 1120 1121
1121 1130 1140 1150 1310 1320 1330 1340 1350 1360 1370 1380
1390

' 101 - 0720 0730 0740 0750 0760 0770 0780

' 102 - 0080 0490 0520 0540 0540 0550 0560 0580 0600 0610 0620 0680
0710 0740 0750 0760 0770 0780 0800 0820 0880 0890 0900 0910
1040 1050 1060 1070 1080 1120 1121 1121 1130 1140 1150 1310
1320 1330 1340 1350 1360 1370 1380 1390 1440 1450 1470 1510
1510 1560 1570 1580 1590 1600 1600 1750 1760 1770 1775 1780
1790 1800 1810 1820 1830 1840 1890 1900 1910 1920 1930 1940
1950 1960 1970 1980 1990 2000 2010 2020 2030 2050 2050 2060
2090 2100 2130 2160 2170 2180 2200 2230 2260

' 105 - 0290 0540 0560 0600 0620 0770 0780 0830 0840 0845 1320 1330
1340 1350 1370 1380 1390 1510 1580 1590 1600 1760 1770 2050
2230

' 124 - 1600 1730 2040

' 222 - 0560 0590 0650

SPECIAL FUNCTION CROSS REFERENCE

' 31 - 2410 2420 2440 2440

' 33 - 2340 2360 2410

' 35 - 2420 2430 2440 2470

' 36 - 2340 2360 2390 2460 2460

' 37 - 2340 2360

' 38 - 2340 2360

' 39 - 2280 2310 2320

' 40 - 2330 2350 2370 2380 2390 2400 2450 2460

' 44 - 2380 2450 2450

' 55 - 2430 2430

' 56 - 2420 2430 2440 2470 2470

' 86 - 2320 2340 2370 2380 2390 2410 2420 2450 2460

' 87 - 2320 2340 2370 2380 2390 2410 2420 2430 2440 2450 2460 2470

' 102 - 2290 2430 2430 2440 2440 2450 2450 2460 2460 2470 2470

' 105 - 2290

MAINTENANCE TAPE CASSETTE LISTING PROGRAM

The maintenance tape cassette listing program DBTAPERD is written in BASIC for the WANG 2200. DBTAPERD requires a single tape cassette drive and 8K of memory to run special function 3. Disk storage capability is required to run special functions 1 and 2.

Special function 1 prints a hard copy listing of the maintenance transactions exactly as they are stored on the cassette tape. After each record is read and printed it is written to the disk file "DB-INPUT." "DB-INPUT" must be cataloged prior to executing S.F. 1. The number of disk sectors required for "DB-INPUT" is a function of the number of maintenance transactions to be written in "DB-INPUT." Each transaction requires one disk sector. A full cassette tape contains approximately 650 transactions.

Special function 2 prints a sequence numbered and formatted hard copy listing of the maintenance transactions contained in disk file "DB-SORT2." "DB-SORT2" should contain the sorted version of "DB-INPUT." Each transaction in the file is sequence numbered. All fields in the transaction are labeled for easy interpretation.

Special function 3 prints a sequence numbered and formatted hard copy listing of the maintenance transaction cassette tape. S.F. 3 is identical to S.F. 2 except that input records are read from the cassette tape rather than from the disk file "DB-SORT2."

DBTAPERD VARIABLE DEFINITIONS	
VARIABLE NAME	DEFINITION
F	Flag used to differentiate between S.F. 2 and S.F. 3 statements. F=1 for S.F. 2; F=0 for S.F. 3.
I	Loop indexing variable used in instruction display. Also used as the transaction sequence counter.
F1$	Alphanumeric conversion of I for printing formatted transaction sequence number.
N	Used to specify block-size equal to 100 bytes in the DATA LOAD BT commands.
T1$	Transaction type.
T2$	Course CDP Number.
T3$	Activity/dept code.
T4$	Maintenance transaction bytes 11-50.
Z	Numeric conversion of the transaction type.
Z$	50 byte maintenance transaction (first 50 bytes of Z1$()).
Z1$()	100 byte maintenance transaction from cassette tape.

DBTAPERD SUBROUTINE DEFINITIONS	
SUBROUTINE NUMBER	DEFINITION
1	Prints an unformatted hard copy of the maintenance transactions and writes the transaction tape on disk file "DB-INPUT."
2	Prints a sequence numbered and formatted hard copy of the sorted transactions from disk file "DB-SORT2."
3	Prints a sequence numbered and formatted hard copy of the maintenance transaction cassette tape.
9	Converts I to I1$; prints the sequence number and the first two fields of each transaction.
13, 14, 15	Used for text entry.

10SEPT76 PAGE 1
PROGRAM LISTING - DBTAPERD

```
  10 DIM Z1$(2)50,Z$50,T1$2,T2$4,T3$4,T4$40,I1$4
  20  PRINT HEX(03)
      :FOR I=1 TO 64
      :PRINT "*";
      :NEXT I
  30 PRINT TAB(15);"DOTS ACTIVITY DATA BASE MAINTENANCE"
      :PRINT
      :PRINT "KEY S.F. 1 TO OBTAIN UNFORMATTED LISTING OF TRANSACTION
      TAPE"
      :PRINT TAB(11);"AND WRITE TAPE TO DISK FILE 'DB-INPUT'."
  40 PRINT "KEY S.F. 2 TO OBTAIN NUMBERED/FORMATTED LISTING OF SORTED"
      :PRINT TAB(11);"TRANSACTIONS FROM DISK FILE 'DB-SORT2'."
  50 PRINT "KEY S.F. 3 TO OBTAIN NUMBERED/FORMATTED LISTING OF TRANSAC
      TION"
      :PRINT TAB(11);"TAPE."
      :PRINT
      :FOR I=1 TO 64
      :PRINT "*";
      :NEXT I
  60  SELECT PRINT 005(64)
      :END
  70 DEFFN'13"NO  NO  DEPT  "
  80 DEFFN'15"STR(Z$,"
  90 DEFFN'14";HEX(2020);"
 100  DEFFN'1
      :PRINT HEX(03)
      :STOP "MOUNT TRANSACTION TAPE CASSETTE, THEN KEY CONTINUE"
      :PRINT HEX(03)
 110  SELECT PRINT 215(64)
 120 DATA LOAD BT (N=100)Z1$()
      :PRINT Z1$(1)
      :IF "END"=STR(Z1$(1),1,3) THEN   60
      :GOSUB  900
      :GOTO  120
 130  DEFFN'2
      :I=1
      :DATA LOAD DC OPEN R "DB-SORT2"
 135  DSKIP 9905
 140  SELECT PRINT 215(64)
 150  F=1
 160  GOTO  190
 170  DEFFN'3
      :I=1
      :F=0
 180  IF F=1  THEN  190
      :DATA LOAD BT (N=100)Z1$()
      :IF "END"=STR(Z1$(1),1,3) THEN   60
      :GOTO  250
 190  DATA LOAD DC T1$,T2$,T3$,T4$
 195  IF END THEN   60
 200  IF T1$="EN"  THEN   60
 210  STR(Z1$(1),1,2)=T1$
```

PROGRAM LISTING

```
220  STR(Z1$(1),3,4)=T2$
230  STR(Z1$(1),7,4)=T3$
240  STR(Z1$(1),11)=T4$
250  CONVERT STR(Z1$(1),1,2) TO Z
     :Z$=Z1$(1)
260  ON ZGOSUB  270 , 280 , 320 , 270 , 270 , 350 , 380 , 410 , 270 ,
     270 , 440 , 470 , 500 , 530 , 270 , 560 , 590 , 620 , 650 , 680
     , 710 , 270 , 740 , 770 , 800 , 830 , 860
     :I=I+1
     :IF Z=0 THEN   60
     :GOTO  180
270  I=I+1
     :GOTO  180
280  PRINT "REC  TR";TAB(33);"FIELD   NEW"
290  PRINT "NO   NO  DEPT  CDP   BLDG  RM    NAME    VALUE";TAB(49);"
     *CHG FAC DATA"
300  GOSUB '9
     :PRINT STR(Z$,7,4);"  ";STR(Z$,11,4);"  ";STR(Z$,15,4);"  ";STR(
     Z$,19,6);"  ";STR(Z$,25,5)
     :PRINT
     :RETURN
310  DEFFN'9
     :CONVERT I TO I1$,(####)
     :PRINT I1$;" ";STR(Z$,1,2);"  ";STR(Z$,3,4);"  ";
     :RETURN
320  PRINT "REC  TR";TAB(15);"INSTRUCTOR   FIELD   NEW"
330  PRINT "NO   NO  DEPT  ID (SSN)     NAME    VALUE";TAB(49);"*CHG
     INST DATA"
340  GOSUB '9
     :PRINT STR(Z$,7,3);"-";STR(Z$,10,2);"-";STR(Z$,12,4);"  ";STR(Z$
     ,16,6);"  ";STR(Z$,22,16)
     :PRINT
     :RETURN
350  PRINT "REC  TR";TAB(21);"FIELD   NEW"
360  PRINT "NO   NO  DEPT  CDP   NAME    VALUE";TAB(49);"*CHG CRS DAT
     A"
370  GOSUB '9
     :PRINT STR(Z$,7,4);HEX(2020);STR(Z$,11,6);HEX(2020);STR(Z$,17,16
     )
     :PRINT
     :RETURN
380  PRINT "REC  TR";TAB(26);"FIELD   NEW"
390  PRINT "NO   NO  DEPT  CDP   SIR  NAME    VALUE";TAB(49);"*CHG CR
     S DATA"
400  GOSUB '9
     :PRINT STR(Z$,7,4);HEX(2020);STR(Z$,11,3);HEX(2020);STR(Z$,14,6)
     ;HEX(2020);STR(Z$,20,4)
     :PRINT
     :RETURN
410  PRINT "REC  TR";TAB(21);"INSTRUCTOR   FIELD  NEW"
420  PRINT "NO   NO  DEPT  CDP   ID (SSN)     NAME   VALUE    *CHG AS
     GN DATA"
```

PROGRAM LISTING

```
 430  GOSUB '9
      :PRINT STR(Z$,7,4);HEX(2020);STR(Z$,11,3);"-";STR(Z$,14,2);"-";
      STR(Z$,16,4);HEX(2020);STR(Z$,20,6);HEX(20);STR(Z$,26,4)
      :PRINT
      :RETURN
 440  PRINT "REC  TR"
 450  PRINT "NO   NO  DEPT  CDP";TAB(49);"*DELETE COURSE"
 460  GOSUB '9
      :PRINT STR(Z$,7,4)
      :PRINT
      :RETURN
 470  PRINT "REC  TR"
 480  PRINT "NO   NO  DEPT  CDP   SIR";TAB(49);"*DELETE SIR"
 490  GOSUB '9
      :PRINT STR(Z$,7,4);HEX(2020);STR(Z$,11,3)
      :PRINT
      :RETURN
 500  PRINT "REC  TR";TAB(15);"INSTRUCTOR"
 510  PRINT "NO   NO  DEPT  ID (SSN)";TAB(49);"*DELETE INSTR"
 520  GOSUB '9
      :PRINT STR(Z$,7,3);"-";STR(Z$,10,2);"-";STR(Z$,12,4)
      :PRINT
      :RETURN
 530  PRINT "REC  TR"
 540  PRINT "NO   NO  DEPT  CDP   BLDG  RM ";TAB(49);"*DELETE FACIL"
 550  GOSUB '9
      :PRINT STR(Z$,7,4);HEX(2020);STR(Z$,11,4);HEX(2020);STR(Z$,15,4)
      ;HEX(2020);STR(Z$,19,4)
      :PRINT
      :RETURN
 560  PRINT "REC  TR";TAB(15);"OLD   NEW"
 570  PRINT "NO   NO  DEPT  CDP   CDP";TAB(49);"*CHG CDP NO"
 580  GOSUB '9
      :PRINT STR(Z$,7,4);HEX(2020);STR(Z$,11,4)
      :PRINT
      :RETURN
 590  PRINT "REC  TR";TAB(21);"OLD  NEW"
 600  PRINT "NO   NO  DEPT  CDP   SIR  SIR";TAB(49);"*CHG SIR"
 610  GOSUB '9
      :PRINT STR(Z$,7,4);HEX(2020);STR(Z$,11,3);HEX(2020);STR(Z$,14,3)
      :PRINT
      :RETURN
 620  PRINT "REC  TR";TAB(15);"OLD INSTR    NEW INSTR"
 630  PRINT "NO   NO  DEPT  ID (SSN)     ID (SSN)";TAB(49);"*CHG ID NO
      "
 640  GOSUB '9
      :PRINT STR(Z$,7,3);"-";STR(Z$,10,2);"-";STR(Z$,12,4);HEX(2020);
      STR(Z$,16,3);"-";STR(Z$,19,2);"-";STR(Z$,21,4)
      :PRINT
      :RETURN
 650  PRINT "REC  TR";TAB(21);"OLD   OLD   NEW   NEW"
 660  PRINT "NO   NO  DEPT  CDP   BLDG  RM    BLDG  RM ";TAB(49);"*CHG
```

PROGRAM LISTING

```
       ROOM NO"
 670   GOSUB '9
       :PRINT STR(Z$,7,4);HEX(2020);STR(Z$,11,4);HEX(2020);STR(Z$,15,4)
       ;HEX(2020);STR(Z$,19,4);HEX(2020);STR(Z$,23,4)
       :PRINT
       :RETURN
 680   PRINT "REC  TR  OLD   NEW"
 690   PRINT "NO   NO  DEPT  DEPT";TAB(49);"*CHC DEPT NO"
 700   GOSUB '9
       :PRINT STR(Z$,11,4)
       :PRINT
       :RETURN
 710   PRINT "REC  TR";TAB(15);"NEW";TAB(32);"COURSE"
 720   PRINT "NO   NO  DEPT  CDP   CIN";TAB(32);"SHORT TITLE";TAB(49);"
       *ADD COURSE"
 730   GOSUB '9
       :PRINT STR(Z$,7,4);HEX(2020);STR(Z$,11,10);HEX(2020);STR(Z$,21,1
       6)
       :PRINT
       :RETURN
 740   PRINT "REC  TR";TAB(21);"NEW  CONT"
 750   PRINT "NO   NO  DEPT  CDP   SIR  HOURS  TP";TAB(49);"*ADD SIR"
 760   GOSUB '9
       :PRINT STR(Z$,7,4);HEX(2020);STR(Z$,11,3);HEX(2020);STR(Z$,14,4)
       ;HEX(202020);STR(Z$,18,1)
       :PRINT
       :RETURN
 770   PRINT "REC  TR";TAB(15);"INSTRUCTOR"
 780   PRINT "NO   NO  DEPT  ID (SSN)     RATE   IN NAME";TAB(49);"*ADD
        INSTR"
 790   GOSUB '9
       :PRINT STR(Z$,7,3);"-";STR(Z$,10,2);"-";STR(Z$,12,4);HEX(2020);
       STR(Z$,16,5);HEX(2020);STR(Z$,21,2);" ";STR(Z$,23,16)
       :PRINT
       :RETURN
 800   PRINT "REC  TR";TAB(31);"RM  RM REQ"
 810   PRINT "NO   NO  DEPT  CDP   BLDG RM   CAP TP HRS   AVHRS *ADD FA
       CIL"
 820   GOSUB '9
       :PRINT STR(Z$,7,4);HEX(2020);STR(Z$,11,4);" ";STR(Z$,15,4);" ";
       STR(Z$,19,3);" ";STR(Z$,22,1);HEX(2020);STR(Z$,23,5);" ";STR(Z$,
       30,5)
       :PRINT
       :RETURN
 830   PRINT "REC  TR";TAB(21);"INSTRUCTOR    %"
 840   PRINT "NO   NO  DEPT  CDP   ID (SSN)     QUAL  ASN";TAB(49);"*AD
       D ASSIGNMT"
 850   GOSUB '9
       :PRINT STR(Z$,7,4);HEX(2020);STR(Z$,11,3);"-";STR(Z$,14,2);"-";
       STR(Z$,16,4);HEX(2020);STR(Z$,20,4);HEX(2020);STR(Z$,24,1)
       :PRINT
       :RETURN
```

PROGRAM LISTING

```
 860  PRINT "REC  TR";TAB(21);"INSTRUCTOR"
 870  PRINT "NO   NO  DEPT  CDP   ID (SSN)";TAB(49);"*DELETE ASSGN"
 880  GOSUB '9
      :PRINT STR(Z$,7,4);HEX(2020);STR(Z$,11,3);"-";STR(Z$,14,2);"-";
      STR(Z$,16,4)
      :PRINT
      :RETURN
 900  T1$=STR(Z1$(1),1,2)
 910  T2$=STR(Z1$(1),3,4)
 920  T3$=STR(Z1$(1),7,4)
 930  T4$=STR(Z1$(1),11)
 940  DATA LOAD DC OPEN R "DB-INPUT"
 950  DSKIP END
 960  DATA SAVE DC T1$,T2$,T3$,T4$
 970  DATA SAVE DC END
 980  REM GET NEXT TAPE RECORD
 990  RETURN
```

10SEPT76 PAGE 1
LINE NUMBER CROSS REFERENCE - **DBTAPERD**

0060 - 0120 0180 0195 0200 0260

0120 - 0120

0180 - 0260 0270

0190 - 0160 0180

0250 - 0180

0270 - 0260 0260 0260 0260 0260 0260 0260

0280 - 0260

0320 - 0260

0350 - 0260

0380 - 0260

0410 - 0260

0440 - 0260

0470 - 0260

0500 - 0260

0530 - 0260

0560 - 0260

0590 - 0260

0620 - 0260

0650 - 0260

0680 - 0260

0710 - 0260

0740 - 0260

0770 - 0260

0800 - 0260

0830 - 0260

0860 - 0260

0900 - 0120

10SEPT76 PAGE 3
VARIABLE CROSS REFERENCE - **DBTAPERD**

```
F      - 0150 0170 0180

I      - 0020 0020 0050 0050 0130 0170 0260 0260 0270 0270 0310

I1$    - 0010 0310 0310

N      - 0120 0180

T1$    - 0010 0200 0210 0900

T2$    - 0010 0190 0220 0910 0960

T3$    - 0010 0190 0230 0920 0960

T4$    - 0010 0190 0240 0930 0960

Z      - 0250 0260 0260

Z$     - 0010 0250 0300 0300 0300 0300 0300 0310 0310 0340 0340 0340
         0340 0340 0370 0370 0370 0400 0400 0400 0400 0430 0430 0430
         0430 0430 0430 0460 0490 0490 0520 0520 0520 0550 0550 0550
         0550 0580 0580 0610 0610 0610 0640 0640 0640 0640 0640 0640
         0670 0670 0670 0670 0670 0700 0730 0730 0730 0760 0760 0760
         0760 0790 0790 0790 0790 0790 0790 0820 0820 0820 0820 0820
         0820 0820 0850 0850 0850 0850 0850 0850 0880 0880 0880 0880

Z1$()  - 0010 0120 0120 0120 0180 0180 0210 0220 0230 0240 0250 0250
         0900 0910 0920 0930
```

10SEPT76 PAGE 4
SPECIAL FUNCTION CROSS REFERENCE

```
' 1 -   0100

' 2 -   0130

' 3 -   0170

' 9 -   0300 0310 0340 0370 0400 0430 0460 0490 0520 0550 0580 0610
       0640 0670 0700 0730 0760 0790 0820 0850 0880

' 13 -  0070

' 14 -  0090

' 15 -  0080
```

MAINTENANCE DATA FILE TRANSMISSION PROGRAM

The maintenance data file transmission program TC-NCSS is written in BASIC for the WANG 2200. TC-NCSS requires a single tape cassette drive and/or a disk storage unit, the General I/O ROM option, a telecommunications modem, and 8K of memory to run.

TC-NCSS was originally written by WANG Laboratories to use the WANG System as a telecommunications terminal to communicate with the NCSS timesharing system. Special function 1Ø was added to create a disk file in the NCSS environment from data stored in the WANG System on either a cassette tape or disk storage unit.

Since TC-NCSS is currently being revised by WANG personnel, only the current (as of 10 Sept 76) program listing is included in this manual.

PROGRAM LISTING - TC-NCSS

```
 10 REM "$TC-NCSS" 10/15/74
     :M=1
     :DIM E1$1,N$1,E$1,R$1,K$1,T$1
     :Q=23
     :L9=180
     :DIM B(23),C(23),Q$1,R$(23,180)1,P$1,W$1
     :R,T=1
     :GOTO  430
 11  REM UPDATED 28 JUL 76
 12  DIM Z$(2)50,T1$2,X2$4,T3$4,T4$40
 20 $IF ON /001, 300
 30 KEYIN T$,  50 , 270
     :IF P$=R$ THEN  20
     :$IF ON #1,  60
     :GOTO  30 :REM KB/TC/PRT
 40 $GIO#1(A200 4000,T2$)R$()<B(Z),C(Z)>
     :W$=HEX(00)
 50 R$(R,T)=T$
     :IF T$=E$ THEN  90
     :PRINT T$;
     :T=T+1
     :GOTO  30
 60 Z=VAL(P$)
     :ADD(P$,01)
     :IF P$<Q$ THEN  70
     :P$=HEX(01)
 70 KEYIN T$,  40 , 290
     :ADD(W$,01)
     :IF W$<HEX(FF) THEN  70
 80 $GIO#1(A200 4000,T2$)R$()<B(Z),C(Z)>
     :W$=HEX(00)
     :GOTO  30
 90 PRINT
     :C(R)=T
     :T,D=1
     :R=R+1
     :IF R<Q THEN  100
     :R=1
100 KEYIN T$, 110 , 280
     :$IF ON /001, 300
     :GOTO  100
110 BIN(R$)=R
     :IF T$=E1$ THEN  130
120 IF T$=N$ THEN  130
     :IF T$=HEX(00) THEN  130
     :GOTO  50
130 KEYIN T$, 120 , 130
     :$IF ON /001, 300
     :GOTO  130
140 T$=HEX(7F)
     :GOTO  50
150 SELECT PRINT 210
```

PROGRAM LISTING

```
      :PRINT K$;
      :SELECT PRINT 005
      :RETURN
 160 KEYIN K$, 180 , 350
 170 IF P$=R$ THEN  160
      :$IF ON #1, 250
      :GOTO  160
 180 IF K$=HEX(08) THEN  200
      :R$(R,T)=K$
      :PRINT K$;
      :IF K$=E$ THEN  210
      :GOSUB  150
      :T=T+1
      :GOTO  160
 190 T=1
      :GOTO  160
 200 T=T-1
      :IF T<1 THEN  190
      :PRINT HEX(082008);
      :R$(R,T)=" "
      :GOSUB  150
      :GOTO  160
 210 IF P$<>R$ THEN  250
      :IF M=2 THEN  240
 220 Z=R
      :ADD(P$,01)
      :IF P$<Q$ THEN  230
      :P$=HEX(01)
 230 $GIO/215(A200 4000,T2$)R$()<B(Z),T>
 240 SELECT INPUT 219
      :GOSUB  150
      :GOTO  310
 250 Z=VAL(P$)
      :ADD(P$,01)
      :IF P$<Q$ THEN  260
      :P$=HEX(01)
 260 $GIO#1(A2004000,T2$)R$()<B(Z),T>
      :GOTO  170
 270 T$=HEX(7F)
      :GOTO  50
 280 T$=HEX(7F)
      :GOTO  110
 290 T$=HEX(7F)
      :GOTO  40
 300 SELECT INPUT 001,PRINT 005
      :GOTO  160
 310 SELECT INPUT 219,PRINT 005,#1215
      :P$,R$=HEX(01)
      :T,R=1
 320 KEYIN T$, 330 , 330
      :$IF ON /001, 300
      :GOTO  320
```

PROGRAM LISTING

```
 330 ON MGOTO  340
      :PRINT T$;
      :GOTO  320
 340 IF T$=E1$ THEN   30
      :IF T$=N$ THEN  320
      :IF T$=HEX(00) THEN  320
      :GOTO   50
 350 ON VAL(K$)-11GOTO  360 , 370 , 420 , 450 , 700
      :PRINT "'";
      :HEXPRINT K$;
      :PRINT "'";
      :GOSUB  150
      :GOTO  160
 360 DEFFN'12
      :M=1
      :SELECT PRINT 005
      :PRINT "KB/TC/CRT/PRT"
      :GOTO  310
 370 DEFFN'13
      :M=2
      :SELECT PRINT 005
      :PRINT "KB/TC/CRT"
      :GOTO  310
 380 DEFFN'14
      :SELECT INPUT 001,PRINT 215
 390 INPUT "0=PAGE EJECT  N=# LINES",Z
      :IF Z=0 THEN  400
      :FOR M=1 TO Z
      :PRINT TAB(1)
      :NEXT M
      :GOTO  410
 400 PRINT HEX(0C)
 410 SELECT PRINT 005
      :RETURN
 420 GOSUB  380
      :GOTO  450
 430 SELECT INPUT 001,PRINT 005
 440 FOR Z=1 TO Q
      :B(Z)=Z*L9-(L9-1)
      :NEXT Z
      :BIN(Q$)=Q
 450 DEFFN'15
      :SELECT PRINT 005(64)
 460 PRINT HEX(03),,"WANG 2200-B TELECOMMUNICATIONS"
 470 IF E$<>" " THEN  480
      :GOSUB  560
 480 PRINT "  KEYBOARD INTERACTION WITH T.C.  AUTO-OUTPUT TO LINE PRIN
      TER."
 490 PRINT "REQUIRES 2200-B (8K)  CRT/KEYBOARD AND 2227 BOARD"
 500 PRINT "      WITH OPTION 2 I/O ROM"
 510 PRINT "S.F.'12 CRT/PRINTER OUTPUT"
 520 PRINT "S.F.'13 CRT OUTPUT"
```

PROGRAM LISTING

```
 530 PRINT "S.F.'14 EJECT PRINTER PAGE"
 540 PRINT "S.F.'15 DISPLAY OF S.F.POSSIBILITIES"
 550 STOP "KEY APPROPRIATE SPECIAL FUNCTION"
 560 PRINT "ENTER END OF LINE SEQUENCE IN HEX, EG."
 570 PRINT " CR/LF/7F    =  0D 0A 7F"
 580 PRINT " CR/LF/00    =  0D 0A 00"
 590 PRINT " LF/CR/7F    =  0A 0D 7F"
 600 GOSUB  610
      :E$=T$
      :GOSUB  610
      :E1$=T$
      :GOSUB  610
      :N$=T$
      :RETURN
 610 GOSUB  620
      :ROTATE(W$,4)
      :AND (W$,F0)
      :T$=W$
      :GOSUB  620
      :AND (W$,0F)
      :OR (T$,W$)
      :PRINT "/";
      :RETURN
 620 KEYIN K$, 630 , 620
      :GOTO  620
 630 W$=K$
      :IF W$<"0" THEN  620
      :IF W$<="9" THEN  640
      :IF W$<"A" THEN  620
      :IF W$>"F" THEN  620
      :ADD(W$,F9)
 640 PRINT K$;
      :RETURN
 700  DEFFN'10
 710  DATA LOAD DC OPEN R "DB-SORT"
 715 DSKIP 377S
 720  INIT(00)T1$,X2$,T3$,T4$
 740  GOSUB  900
 745  IF T1$="EN"  THEN  800
 750  SELECT PRINT 21D
 760 FOR L=1 TO 150
      :NEXT L
 770      PRINT Z$(1);
 785  PRINT HEX(0D0A);
 787  SELECT PRINT 215(64)
      :PRINT Z$(1);HEX(0D0A)
 790  SELECT INPUT 219,PRINT 215, #1215
      :P$,R$=HEX(01)
      :T,R=1
 791  KEYIN T$, 792 , 792
 792  R$(R,T)=T$
      :HEXPRINT T$;
```

PROGRAM LISTING

```
      :T=T+1
      :GOTO  794
 793  $IF ON /219,  740
 794  KEYIN T$, 792 , 792
      :IF P$=R$ THEN  793
      :GOTO  794
 800  SELECT PRINT 210
      :PRINT HEX(0D0A0D0A),"FILE"
      :GOTO  310
 900  REM
 910  DATA LOAD DC T1$,X2$,T3$,T4$
 912  IF T1$="EN"  THEN  960
 915  IF END THEN  960
 920  STR(Z$(1),1,2)=T1$
 930  STR(Z$(1),3,4)=X2$
 940  STR(Z$(1),7,4)=T3$
 950  STR(Z$(1),11,40)=T4$
 960  RETURN
```

LINE NUMBER CROSS REFERENCE - TC-NCSS

0020 - 0030

0030 - 0030 0050 0080 0340

0040 - 0070 0290

0050 - 0030 0120 0140 0270 0340

0060 - 0030

0070 - 0060 0070

0090 - 0050

0100 - 0090 0100

0110 - 0100 0280

0120 - 0130

0130 - 0110 0120 0120 0130 0130

0150 - 0180 0200 0240 0350

0160 - 0170 0170 0180 0190 0200 0300 0350

0170 - 0260

0180 - 0160

0190 - 0200

0200 - 0180

0210 - 0180

0230 - 0220

0240 - 0210

0250 - 0170 0210

0260 - 0250

0270 - 0030

0280 - 0100

0290 - 0070

0300 - 0020 0100 0130 0320

LINE NUMBER CROSS REFERENCE

0310 - 0240 0360 0370 0800

0320 - 0320 0330 0340 0340

0330 - 0320 0320

0340 - 0330

0350 - 0160

0360 - 0350

0370 - 0350

0380 - 0420

0400 - 0390

0410 - 0390

0420 - 0350

0430 - 0010

0450 - 0350 0420

0480 - 0470

0560 - 0470

0610 - 0600 0600 0600

0620 - 0610 0610 0620 0620 0630 0630 0630

0630 - 0620

0640 - 0630

0700 - 0350

0740 - 0793

0792 - 0791 0791 0794 0794

0793 - 0794

0794 - 0792 0794

0800 - 0745

0900 - 0740

0960 - 0912 0915

VARIABLE CROSS REFERENCE

```
A2     - 0040 0080 0230 0260

B()    - 0010 0040 0080 0230 0260 0440

C()    - 0010 0040 0080 0090

D      - 0090

E$     - 0010 0050 0180 0470 0600

E1$    - 0010 0110 0340 0600

F0     - 0610

F9     - 0630

K$     - 0010 0150 0160 0180 0180 0180 0180 0350 0350 0620 0630 0640

L      - 0760 0760

L9     - 0010 0440 0440

M      - 0010 0210 0330 0360 0370 0390 0390

N$     - 0010 0120 0340 0600

P$     - 0010 0030 0060 0060 0060 0060 0170 0210 0220 0220 0220 0250
         0250 0250 0250 0310 0790 0794

Q      - 0010 0090 0440 0440

Q$     - 0010 0060 0220 0250 0440

R      - 0010 0050 0090 0090 0090 0090 0090 0110 0180 0200 0220 0310
         0790 0792

R$     - 0010 0030 0110 0170 0210 0310 0790 0794

R$()   - 0010 0040 0050 0080 0180 0200 0230 0260 0792

T      - 0010 0050 0050 0050 0090 0090 0180 0180 0180 0190 0200 0200
         0200 0200 0230 0260 0310 0790 0792 0792 0792

T$     - 0010 0030 0050 0050 0050 0070 0100 0110 0120 0120 0130 0140
         0270 0280 0290 0320 0330 0340 0340 0340 0600 0600 0600 0610
         0610 0791 0792 0792 0794

T1$    - 0012 0720 0745 0912 0920

T2$    - 0040 0080 0230 0260

T3$    - 0012 0720 0910 0940
```

10SEPT76 PAGE 5
VARIABLE CROSS REFERENCE

```
T4$    - 0012 0720 0910 0950

W$     - 0010 0040 0070 0070 0080 0610 0610 0610 0610 0610 0630 0630
         0630 0630 0630 0630

X2$    - 0012 0720 0910 0930

Z      - 0040 0040 0060 0080 0080 0220 0230 0250 0260 0390 0390 0390
         0440 0440 0440 0440

Z$()   - 0012 0770 0787 0920 0930 0940 0950
```

10SEPT76 PAGE 6
SPECIAL FUNCTION CROSS REFERENCE

```
' 10 -  0700

' 12 -  0360

' 13 -  0370

' 14 -  0380

' 15 -  0450
```

NCSS MAINTENANCE TRANSACTION PROCESSOR

The general flow of all of the maintenance items in the DOTS system are in Figure I-1. The specific transaction types are presented earlier in this section in Table V-1.

All maintenance transactions originate at the WANG computer. For a given maintenance run, all transactions are accumulated onto a single transaction tape. Each transaction is identified by a format number; the transactions appear on the tape in sorted order by this format number. A specific maintenance run may be made with one or an indefinite number of transactions of any one or all formats.

Once the tape has been transferred to the NCSS system, the transactions reside in a specific NCSS file identified by a filename and filetype. To initiate the action which will ultimately result in presentation of the transaction data to the RAMIS system for execution, two executive sequences are significant. The executives (SUPER and SUPERX) have been provided to control the software at the NCSS computer. The usage of SUPER and SUPERX was covered in Section II of this document and will not be reproduced here.

An elaborate method of accomplishing RAMIS meintenance within the NCSS system was adopted to obviate the necessity of requiring that the system users be highly trained in the details of the RAMIS system. The major design criteria has been to accomplish the maximum amount of maintenance while requiring a minimum amount of data to be user-provided.

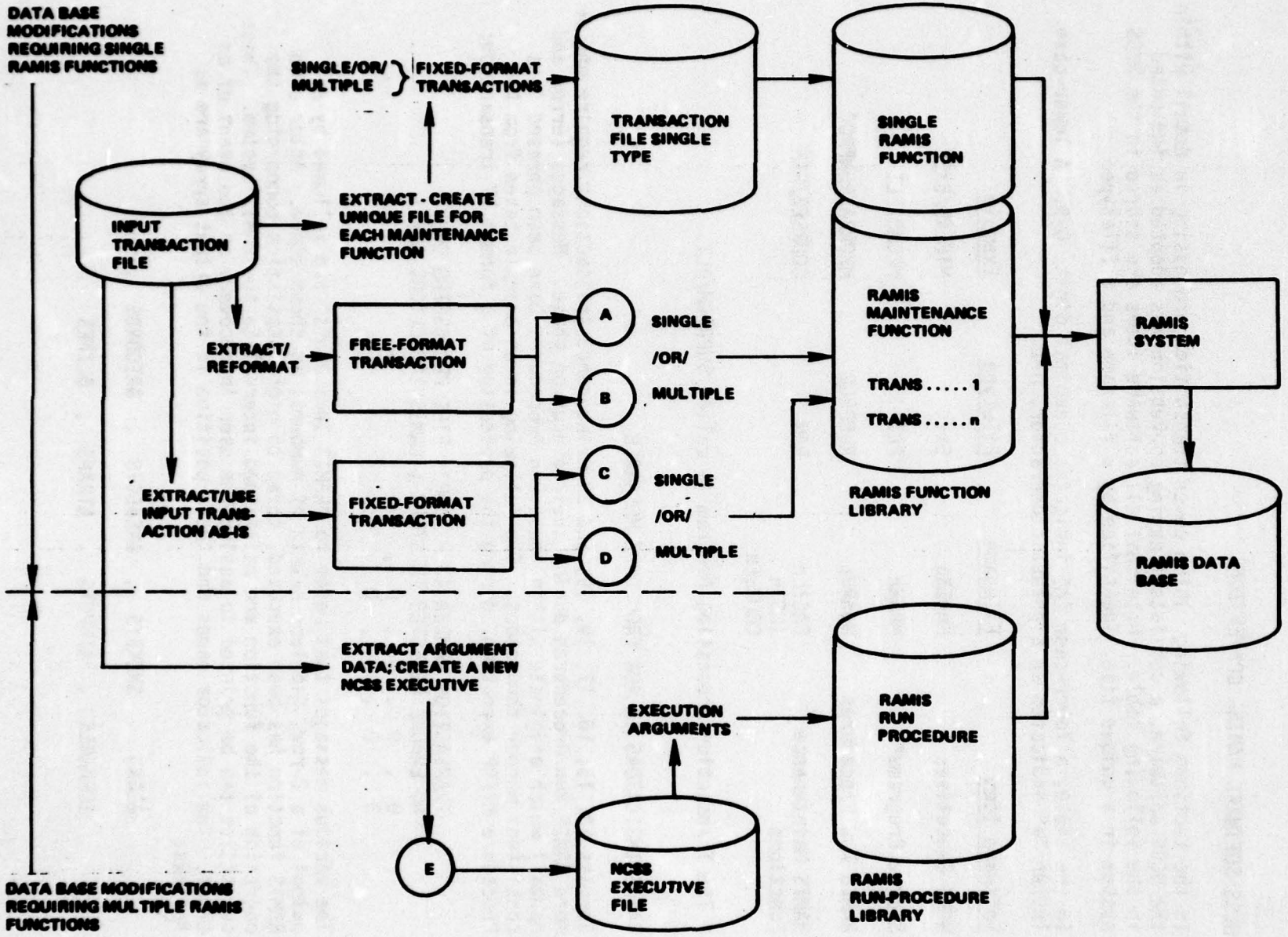
Table V-1 makes reference to "single/multiple step" maintenance. A specific maintenance objective is considered single step if all of the information required for processing has been provided by the user. As an example, consider the change of a length field for a certain training course. The user must provide the course CDP number to identify the course of concern; the name of the length field and the new lengh value are also obtained directly from the user. During NCSS processing, this data can be accumulated into a single RAMIS UPDATE function; immediate processing is possible.

Transaction formats 13, 15, 16, 17, 18, 19, and 20 are tagged as multiple step formats in Table V-1. Consider a format 20 operation which causes a change in an existing school code in all three DOTS files. School code is not a key field in either of the three files. To effect the change in the Course or Facilities files, the course CDP numbers of all of the records of the existing school must be located. For the instructor file, the instructor ID's of all records tagged with the existing school must be located. Thus, change of an existing school code results in two distinct RAMIS functional steps for each file. First, all records bearing the original school code are located and the record highest level key field value is written into a RAMIS spinoff utility file. Second, the key values saved in the spinoff file are used as input to a RAMIS CHANGE function to direct school code change in the required records. These six RAMIS steps are accomplished while requiring the user to provide only the old and the new school codes to initiate the operation.

Figure V-4 is a pictorial presentation of the methods that were considered in the implementation of the RAMIS maintenance activities for the DOTS system. Each indicates transaction input data being extracted from a file and reformatted into a form suitable for RAMIS system input. The processing of the input transaction file is accomplished by a PL/1 program called SUPRPGM.

The maintenance logic indicated at the top of Figure V-4 was not implemented as an indeterminate number of RAMIS function libraries (maintenance utility files on Figure I-1) would be required.

The paths of logic indicated in circles on Figure V-4 (tagged with letters "A" through "E", inclusive) indicate the routes chosen for transaction processing. In the sections following, each transaction format is individually covered in detail; in these sections the type (A-E) is indicated for each unique transaction format.



RAMIS MAINTENANCE OPERATIONS

FIGURE V-4

NCSS SOFTWARE NAMING CONVENTIONS

In the sections following, which cover transaction processing in detail within the NCSS software, a consistent naming convention was adopted as indicated in the following table. Note that all software items are stored in the NCSS system in a unique file, identified by a filename and a filetype:

In the table, a lower-case "n" indicates a decimal digit, 0-9. A lower-case letter "a" indicates an alphabetic character, A-Z.

Software Item	Filename	Filetype	Example
NCSS Executives	MnnEXa	EXEC	M18EXB/EXEC
Source Programs*	MPGMn	PLI	MPGM3/PLI
RAMIS Run Procedures	MnnRPa	RUNPROC	M20RPA/RUNPROC
RAMIS Maintenance Functions	FACILn INSTn COURSEn	RAM	COURSE2/RAM

* *The Transaction Processing Program is called SUPRPGM/PLI.*

RAMIS INDICATORS IN RUN PROCEDURE MESSAGES

Formats 13, 15, 16, 17, 18, 19, and 20 maintenance transactions require one or more RAMIS Run Procedures during their execution phase. Messages (error and normal) which originate within these Run Procedures have been phrased in a consistent manner throughout. The following is an example taken from Run Procedure M17RP execution, during the processing of a format 17 transaction:

```
TRANSACTION NUMBER 43 - COURSE FILE PROCESSING ONLY
NO ERRORS DETECTED IN STEP3 ; RAMIS INDICATORS ARE :
0 , 0 , 0 , 0 ,
3 , 0 , 3 , 0
```

The various messages that refer to "RAMIS Indicators" are followed by an output of a 2-row, 4-column matrix of numbers, as shown above. After each RAMIS function has been executed, three to eight statistics concerning the operation of the function are maintained internal to the RAMIS system. These quantities can be printed to assist the user in recovery in the event of an error. The indicator names and their position in the output array are as follows:

```
&COND , &NOKEYS , &REJECTS , &RECORDS ,

&CHANGES , &INPUTS , &TRANS , &LINES
```

The meaning associated with these internal statistical indicators is explained below:

&COND - Completion indicator; zero (0) if the activity completed normally; (16) if an error occurred.

&NOKEYS - The total number of records rejected with a NOKEY error indication.

&REJECTS - The total number of records rejected with FORMAT, UPPER, or LOWER bound errors.

&RECORDS - The number of records retrieved in TABLE, TABLEF, GRAPH, DUMP, or REPORT mode. If the previous function was not one of these, this indicator will be zero.

&CHANGES - The total number of records that were changed, updated, or deleted.

&INPUTS - The number of new records input to the RAMIS file.

&TRANS - When in data management mode, the number of logical input transactions processed.

&LINES - For TABLE and GRAPH Modes, the number of lines of output generated. When in data management mode, this indicator is an accumulation of the number of miscellaneous rejected records which use the standard system format.

NCSS MAINTENANCE TRANSACTION PROCESSING PROGRAMS

TRANSACTION FORMAT : 2

FUNCTION : Second Level Non-Key Field Changes to Facilities File

DATA BASE(S) MODIFIED : Facilities File

TRANSACTION TYPE : General Type B
Single Step RAMIS Execution

TRANSACTION FILE : Filename/Filetype = DSC/RAM

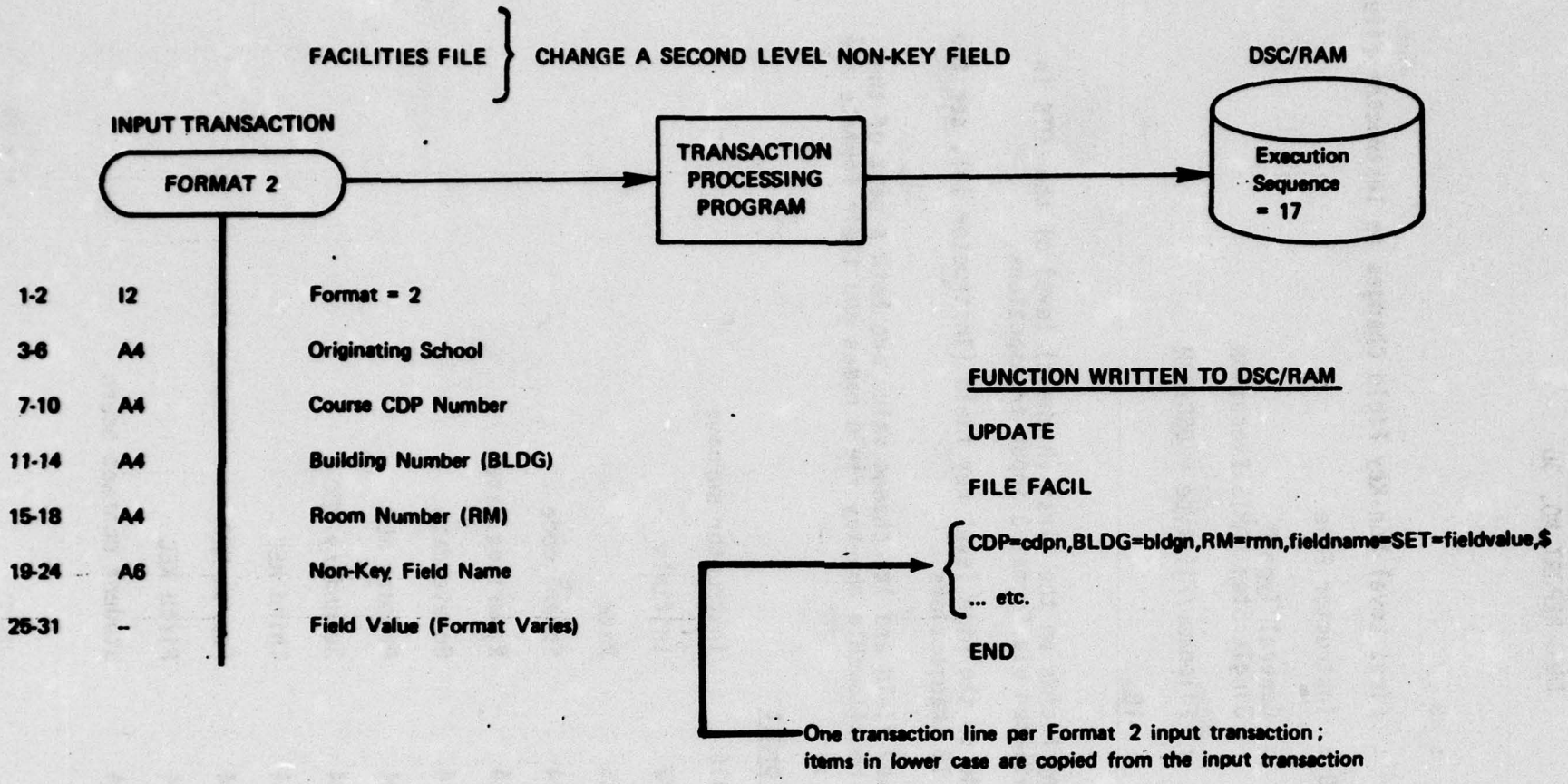
EXECUTION SEQUENCE : 17

This transaction serves to effect any changes required in non-key fields of the second level (Building/Room) segment of the DOTS Facilities file.

For changes required in the key fields of the lower level segment, i.e., building or room, see transaction Format 19.

The name of the field to be modified is carried along in columns 19-24 of the input transaction. The allowable non-key field names are:

Field Name	Format	
RMCAP	I3	Room capacity
RMTP	A1	Room type
REQHRS	F6.1	Required hours
AVHRS	I5	Available hours



FORMAT 2 INPUT TRANSACTION PROCESSING

TRANSACTION FORMAT : 3

FUNCTION : First Level Non-Key Field Changes to Instructor File

DATA BASE(S) MODIFIED : Instructor File

TRANSACTION TYPE : General Type B
Single Step RAMIS Execution

TRANSACTION FILE : Filename/Filetype = DSC/RAM

EXECUTION SEQUENCE : 18

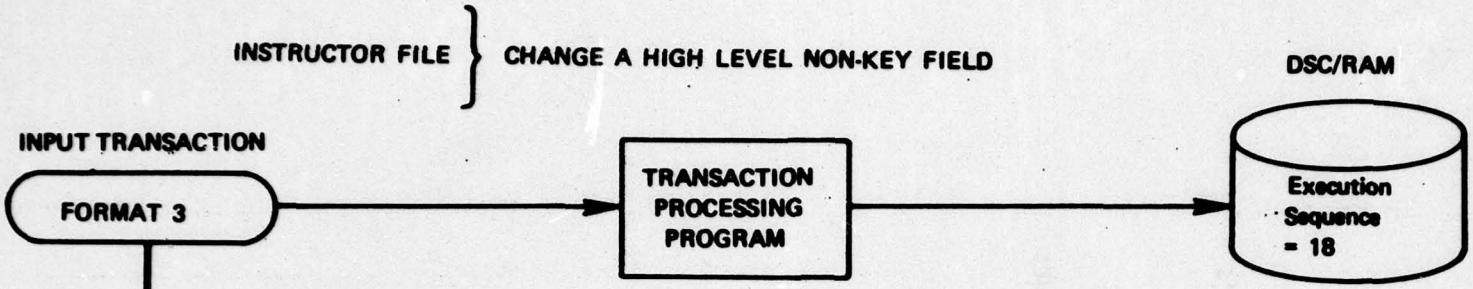
All changes to non-key fields on the first (highest) level of the DOTS Instructor file are initiated via Format 3 input transactions.

For changes to be made to the high level key field (instructor ID), see the discussion of Format 18 transactions.

The name of the non-key field and its change value are both a part of the input transaction. The allowable non-key field names and their formats are as follows:

Field Name	Format	
NME	A16	Instructor surname
IN	A2	Initials
RATE	A5	Rate
SCH	A4	School code
REP	I4	Reporting date
ROT	I4	Rotate date
PNEC	A4	Primary NEC
SNEC	A4	Secondary NEC
NEC3	A4	Third NEC
NEC4	A4	Fourth NEC
NEC5	A4	Fifth NEC
CONT	I4	Student contact hours

Field Name	Format	
CDEV	I4	Curriculum development hours
EQMT	I4	Equipment maintenance hours
PREP	I4	Class preparation hours
BKIN	I4	Instructor break-in hours
CTRN	I4	Instructor cross-training hours
FTRN	I4	Factory training hours
MDUT	I4	Military duty hours
ISCH	I4	Instructor training school hours
FLET	I4	Fleet assistance hours
ADMN	I4	Administrative hours
ANLV	I4	Annual leave hours
SKLV	I4	Sick leave hours
SLIB	I4	Special liberty hours



1-2	I2	Format = 3
3-6	A4	Originating School
7-15	I9	Instructor ID (Social Security No.)
16-21	A6	Field Name
22-37	A16	Field Value

FUNCTION OUTPUT TO DSC/RAM

UPDATE

FILE INST

ID=idnumber,fieldname=SET=fieldvalue,$

...etc.

END

One transaction line per Format 3 input transaction; items in lower case are copied from the input transaction

FORMAT 3 INPUT TRANSACTION PROCESSING

TRANSACTION FORMAT : 6

FUNCTION : Non-Key Changes to the High Level Segment of the Course File

DATA BASE(S) MODIFIED : Course File

TRANSACTION TYPE : General Type B
Single Step RAMIS Execution

TRANSACTION FILE : Filename/Filetype = DSC/RAM

EXECUTION SEQUENCE : 19

Format 6 input transactions are used to cause change to non-key fields within the high level segment of a Course file record.

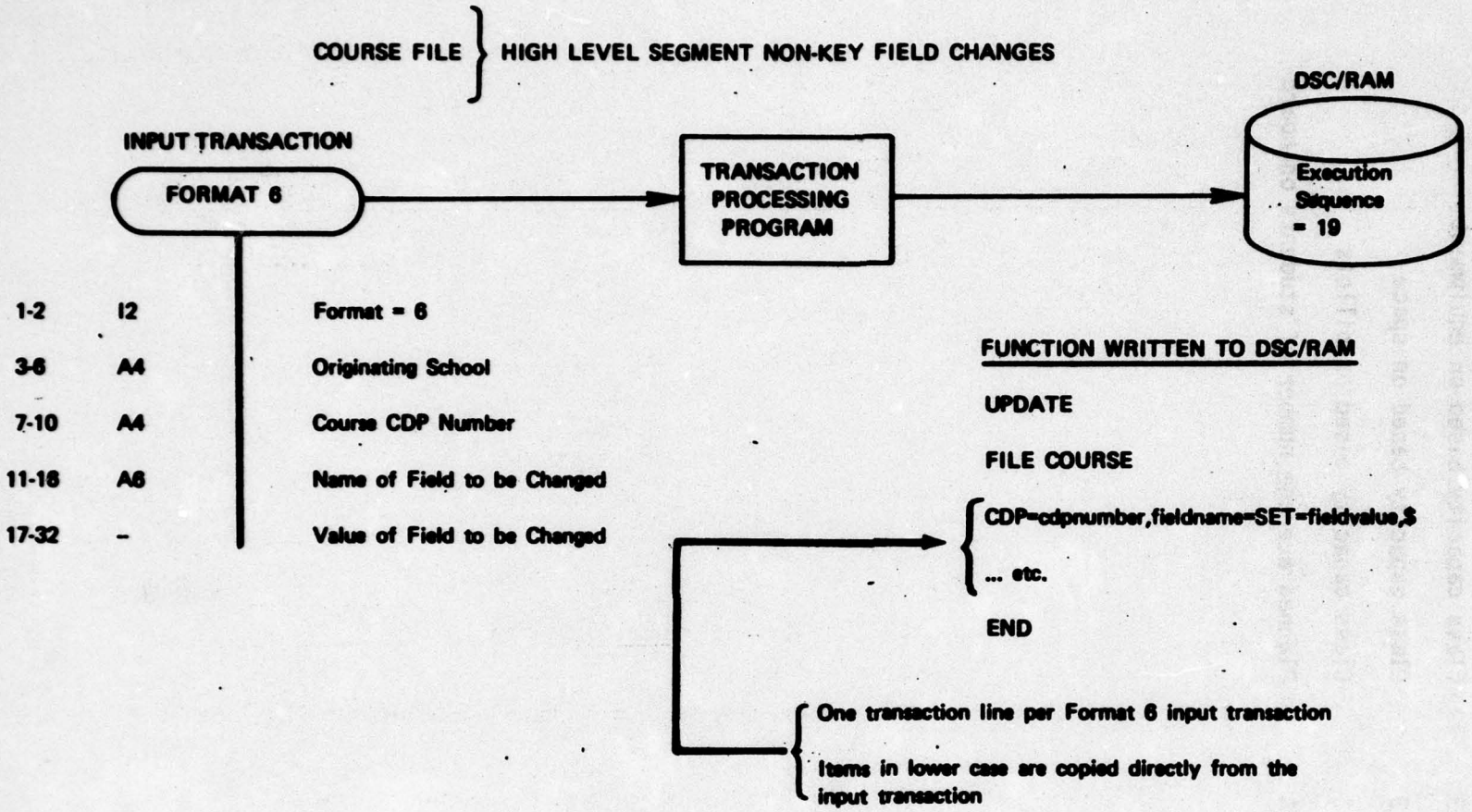
The key field of the high level segment (course CDP number) is changed via Format 16 input transactions.

Within a single Format 6 transaction is included both the name of the non-key field subject to change and its change value. The table following indicates the legal choices for non-key fields and their formats:

Field Name	Format	
SCH	A4	School code
CTLG	A10	Course CIN number
CRS	A16	Course title
CONV1	I3	CFY convening frequency
LEN1	I4	CFY course length
CAP1	I3	CFY class capacity
BCAP1	I3	CFY BUPERS capacity
DMD1	I5	CFY annual demand
BDMD1	I5	CFY BUPERS annual demand
PFAIL	F5.3	Failure rate
NADIS	F5.3	Non-academic disenrollment rate
NOSHOW	F5.3	No-show rate

Field Name	Format	
SETB	F5.3	Setback rate
BKLG	I4	Backlog weeks
MAXCAP	I4	Maximum quota for one class
CONV2	I3	CFY+1 convening frequency
LEN2	I4	CFY+1 course length
CAP2	I3	CFY+1 class capacity
BCAP2	I3	CFY+1 BUPERS class capacity
DMD2	I5	CFY+1 annual demand
BDMD2	I5	CFY+1 BUPERS annual demand
CONV3	I3	CFY+2 convening frequency
LEN3	I4	CFY+2 course length
CAP3	I3	CFY+2 class capacity
BCAP3	I3	CFY+2 BUPERS class capacity
DMD3	I5	CFY+2 annual demand
BDMD3	I5	CFY+2 BUPERS annual demand
MINCAP	I4	Minimum number quotas for one class
PRTY	I2	Training priority designator code
CTYPE	A2	NITRAS class type
PNEC	A4	Primary NEC awarded
UCAP	I3	Course utilization
RDTE	I6	Date next curriculum review
SUIC	A5	Staff unit identification code
CCMM	A10	Course curricula model manager
OPT1	A6	Optional use field
OPT2	A6	Optional use field
OPT3	A6	Optional use field

Field Name	Format	
ECAP	I3	Class capacity based on equipment
SCAP	I3	Class capacity based on space
PCAP	I3	Class capacity based on billets
PAOB	I4	Planned average number of students on-board



FORMAT 6 INPUT TRANSACTION PROCESSING

TRANSACTION FORMAT : 7

FUNCTION : Non-Key Field Changes to Student/Instructor Ratio (SIR) Segment in the Course File

DATA BASE(S) MODIFIED : Course File

TRANSACTION TYPE : General Type B
Single Step RAMIS Execution

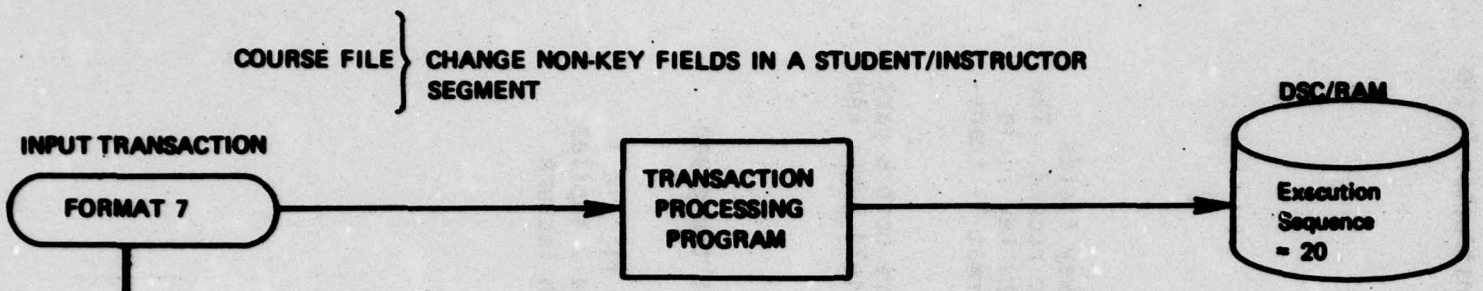
TRANSACTION FILE : Filename/Filetype = DSC/RAM

EXECUTION SEQUENCE : 20

Format 7 input transactions introduce changes into the non-key fields of a student/instructor ratio (SIR) segment of a Course file record. The SIR segment is the second level segment; modifications to the key field of the SIR segment (the SIR, itself) are accomplished by Format 17 transactions.

The actual name of the non-key field and its change value are both a part of the Format 7 input transaction. The allowable non-key field names and their formats are as follows:

Field Name	Format	
F2	I4	Instructional contact hours at the given SIR ratio
F3	I1	Theory/laboratory instruction flag; a 1 indicates a lecture session, a 2 implies laboratory; a 3 indicates both lecture and laboratory



1-2	I2	Format = 7
3-6	A4	Originating School
7-10	A4	Course CDP Number
11-13	I3	Student/Instructor Ratio (SIR)
14-19	A6	Name of Field to be Changed
20-23	–	Field Change Value (Format Varies)

FUNCTION WRITTEN TO DSC/RAM

UPDATE

FILE COURSE

CDP=cdpnumber,F1=sirvalue,fieldname=SET=fieldvalue,$

... etc.

END

- One transaction line per Format 7 input transaction
- Items in lower case are copied from the input transaction
- SIR is known by the name 'F1' in the Course file

FORMAT 7 INPUT TRANSACTION PROCESSING

TRANSACTION FORMAT : 8

FUNCTION : Non-Key Field Changes of Instructor Segment in the Course File

DATA BASE(S) MODIFIED : Course File

TRANSACTION TYPE : General Type B
Single Step RAMIS Execution

TRANSACTION FILE : Filename/Filetype = DSC/RAM

EXECUTION SEQUENCE : 21

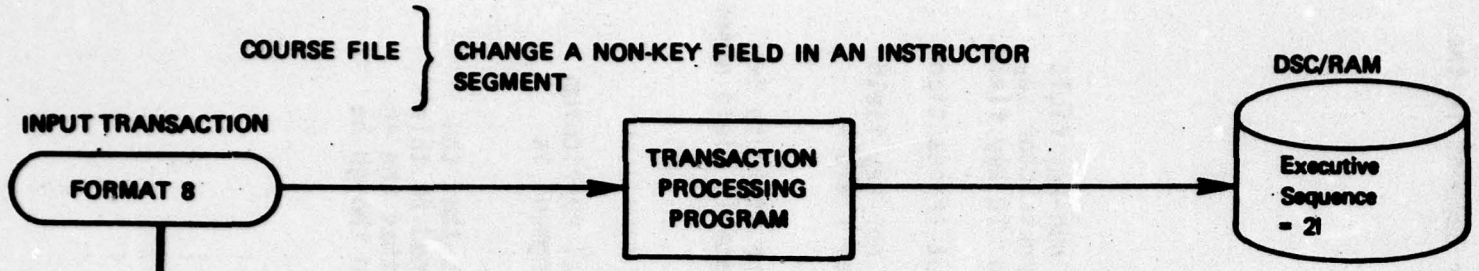
A Format 8 input transaction serves to effect a change to a non-key field within an instructor segment of a Course file record. The instructor segment is the lower level segment, controlled by the instructor ID key field.

Changes to instructor ID are accomplished via Format 18 input transactions.

The action of a Format 8 operation produces a change in the non-key field only of the lower level segment. The high level segment (key field = course CDP number) remains unchanged.

Within the Format 8 transaction are contained the name of the field to be changed and its change value. The following lists the acceptable field names and their formats:

Field Name	Format	
F2	I4	The portion (in %) of the total curriculum that the instructor of this segment is qualified to teach
F3	I1	Assignment flag; a 1 indicates that the instructor is currently assigned to this course; a 0 (zero) indicates that the instructor is not assigned, even though he may be qualified



1-2	I2	Format = 8
3-6	A4	Originating School
7-10	A4	Course CDP Number
11-19	I9	Instructor ID Number (Social Security No.)
20-25	A6	Name of the Field to Change
26-29	-	Value of the Changed Field (Format Varies)

FUNCTION WRITTEN OUT TO DSC/RAM

UPDATE

FILE COURSE

CDP=cdpnumber,F1=instructorid,fieldname=SET=fieldvalue,$

... etc.

END

- One transaction line per Format 8 input transaction
- Items in lower case are copied directly from the input transaction
- Instructor ID is known by the name 'F1' in the Course file

FORMAT 8 INPUT TRANSACTION PROCESSING

TAEG REPORT NO. 36

TRANSACTION FORMAT : 11

FUNCTION : Delete a Course

DATA BASE(S) MODIFIED : Course File

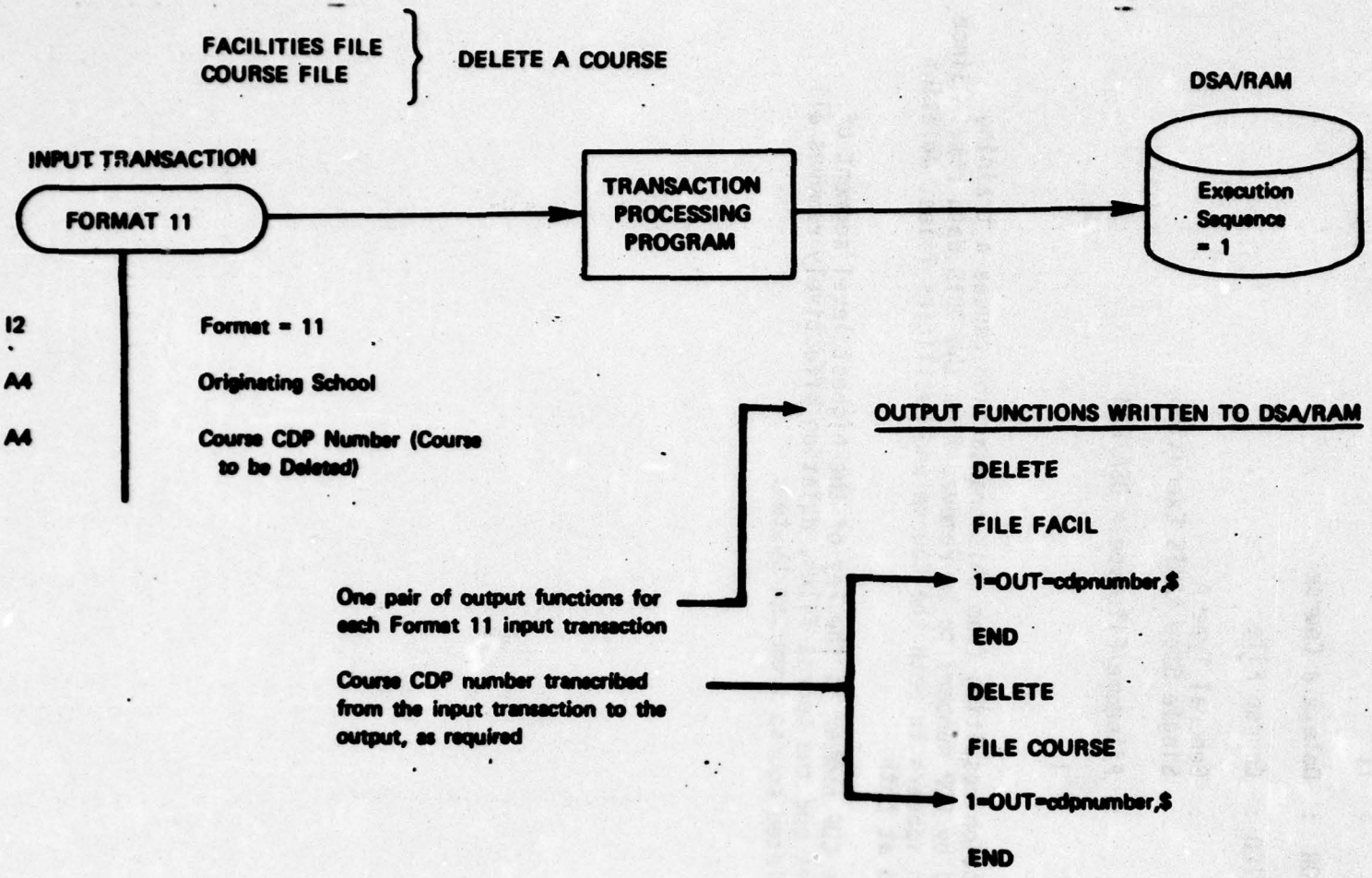
TRANSACTION TYPE : General Type A
Single Step RAMIS Execution

TRANSACTION FILE : Filename/Filetype = DSA/RAM

EXECUTION SEQUENCE : 1

The maintenance action resulting from this transaction causes a training course (identified by CDP number) to be removed from the DOTS data base. Since course information appears in both the Course and Facilities files, deletion action is directed at both.

Inasmuch as course CDP number is the key of the highest level segment of both the Facilities and the Course files, deletion effectively removes all mention of the deleted course from the system.



FORMAT 11 INPUT TRANSACTION PROCESSING

TRANSACTION FORMAT : 12

FUNCTION : Delete a Student/Instructor Ratio (SIR) Segment from the Course File

DATA BASE(S) MODIFIED : Course File

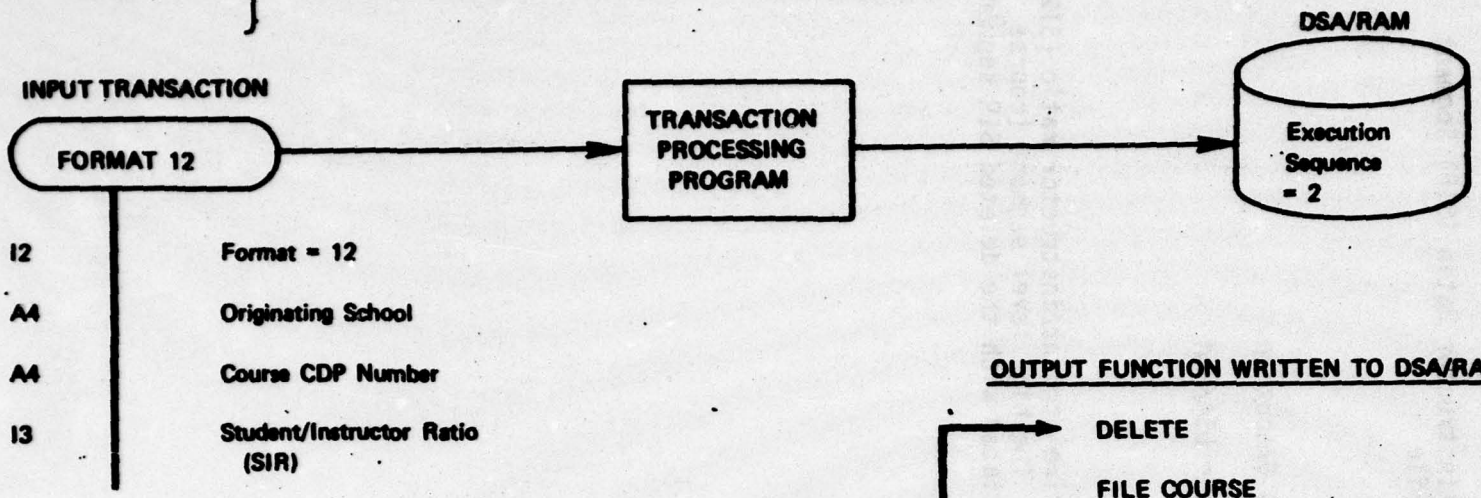
TRANSACTION TYPE : General Type A
Single Step RAMIS Execution

TRANSACTION FILE : Filename/Filetype = DSA/RAM

EXECUTION SEQUENCE : 2

This transaction forces deletion of an entire student/instructor ratio (SIR) lower level segment from the Course file. The high level segment (course CDP controls the high level segment) associated with the deleted SIR segment remains unchanged.

COURSE FILE } DELETE A STUDENT/INSTRUCTOR RATIO SEGMENT



1-2	I2	Format = 12
3-6	A4	Originating School
7-10	A4	Course CDP Number
11-13	I3	Student/Instructor Ratio (SIR)

OUTPUT FUNCTION WRITTEN TO DSA/RAM

DELETE

FILE COURSE

CDP=cdpnumber,F1=OUT=sirnumber,$

END

NOTES

(1) One DELETE function for each Format 12 input transaction

(2) Items in lower case are transcribed from the input transaction

(3) Student/instructor ratio is known by the name 'F1' in the Course file

FORMAT 12 INPUT TRANSACTION PROCESSING

AD-A040 238 IBM CORP CAPE CANAVERAL FLA F/G 5/9
DESIGN OF TRAINING SYSTEMS. TRAPAC USER'S MANUAL, DATA BASE, SC--ETC(U)
SEP 76 K V BRANCH, J D STALEY, R E YANKO N61339-73-C-0097
UNCLASSIFIED TAEG-36 NL

4 OF 6
AD A040238



TRANSACTION FORMAT : 13

FUNCTION : Delete an Instructor

DATA BASE(S) MODIFIED : Instructor File
Course File

TRANSACTION TYPE : General Type E
Multiple Step RAMIS Execution

TRANSACTION FILE : Filename/Filetype = M13EXA/EXEC

EXECUTION SEQUENCE : 5

ADDITIONAL SUPPORT : (1) M13EXB/EXEC - NCSS Executive
(2) M99EXA/EXEC - NCSS Executive
(3) M13RP/RUNPROC - RAMIS Run Procedure

Processing of this transaction type causes the deletion of an instructor from the DOTS data base. The input transaction specifies the instructor ID (Social Security Number) as the key which controls the deletion.

The deletion operation is universal over both the Instructor and Course files, i.e., all mention of the given instructor is removed. The deletion operation proceeds in multiple steps, each requiring the execution of a distinct RAMIS maintenance function.

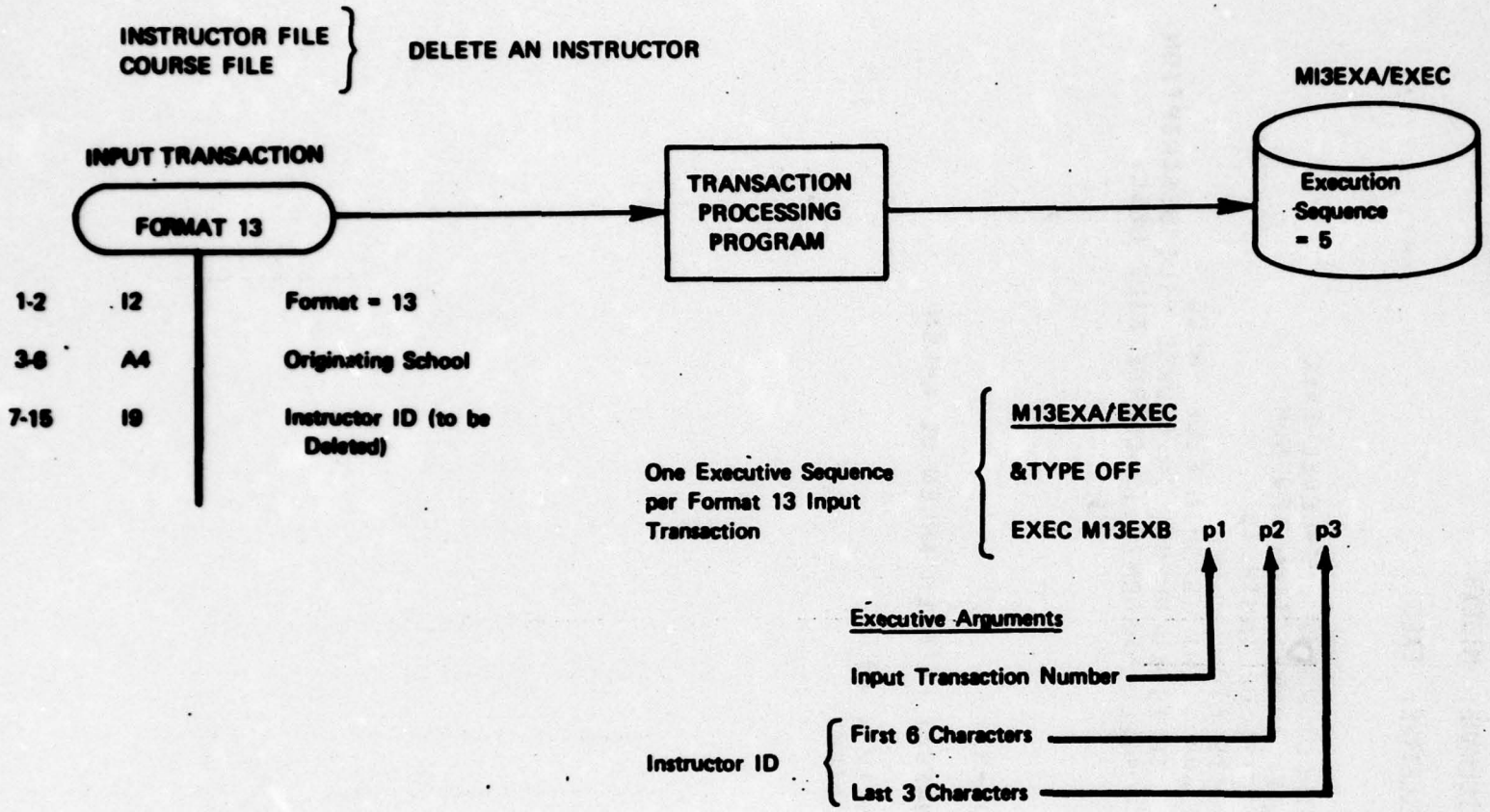
The three RAMIS maintenance functions required to support this transaction type are included in the RAMIS Run Procedure M13RP. The sequence used to invoke this operation is as follows:

(1) The Input Transaction Processing Program constructs an NCSS executive file (M13EXA/EXEC) in response to a Format 13 input transaction. A separate executive sequence is formed for each unique Format 13 input. Execution of a M13EXA executive causes the execution of executive M13EXB. The transaction number and the instructor ID are passed as arguments.

(2) Executive M13EXB performs the required RAMIS system initialization and invokes the RAMIS Run Procedure M13RP. M13RP will be run one time for each Format 13 input transaction. The input transaction number and the ID of the instructor to be deleted are passed as arguments between M13EXB and M13RP.

In the Instructor file, instructor ID is the key field on the highest level. Deletion of the instructor ID in this instance removes the entire record, effectively removing all mention of the given instructor from the file. Since ID is the file highest level key, a single RAMIS DELETE function is required. STEP1 of the Run Procedure M13RP accomplishes this deletion.

In the Course file, instructor ID is the key field controlling the second level. Course CDP number is the highest level key. In order to locate all occurrences of the given ID at the lower level, it is necessary to find all of the courses associated with the given instructor. STEP2 of Run Procedure M13RP produces a RAMIS spinoff file containing the CDP numbers of all Courses of concern. This spinoff file is subsequently used in STEP3 of M13RP to control a standard RAMIS DELETE function directed at the Course file.



FORMAT 13 INPUT TRANSACTION PROCESSING

FORMAT 13 PROCESSING: NCSS EXECUTIVE = M13EXB

FILENAME: M13EXB

FILETYPE: EXEC

```
    &TYPE OFF
    &COMMENT DELETE AN INSTRUCTOR ID, 2ND-LEVEL EXEC
    &COMMENT ENTRY ARGS : (1) TRANSACTION SEQ NUM
    &COMMENT (2) FIRST 6 DIGITS OF INSTR ID
    &COMMENT (3) LAST 3 DIGITS ETC.
    &COMMENT INTERNAL ARGS ARE : (R) IS SPIN FILE LRECL
    &COMMENT (IDN) IS INSTR ID FIELD NUMBER IN COURSE FILE DESCRIPTION
    &COMMENT (IDF) IS INSTR FIELD MNEMONIC IN COURSE FILE DESC.
    &RLEN = 13
    &IDN = 53
    &IDF = F1
STATE FILE FT04F001 P
    &IF &INDEX0 NE 0 &GOTO -S1
ERASE FILE FT04F001 P
-S1 FILEDEF 4 DSK FILE FT04F001 RE F LR &RLEN BL &RLEN
    &STACK KT
    &STACK EXEC M13RP &1 , &2 , &3 ,
    &STACK &IDF , &RLEN , &IDN
    &STACK FIN
ATTACH RAMIS ALL BR
FILEDEF 6 MSG KILL
EXEC RAMIS
DETACH RAMIS BR
    &STACK RT
FILEDEF FT04F001 CLEAR
FILEDEF 6 CONO
    &STACK KT
    &STACK L. /SPACE/
    &STACK C /SPACE//
    &STACK DOWN
    &STACK DOWN
    &STACK L. /SPACE/
    &STACK C /SPACE//
    &STACK DOWN.
    &STACK P 3
    &STACK L. /SPACE/
    &STACK C /SPACE//
    &STACK DOWN.
    &STACK P 3
    &STACK L. /SPACE/
    &STACK C /SPACE//
    &STACK DOWN.
    &STACK P 3
    &STACK Q
EDIT MSG KILL
EXEC M99EXA &1
    &EXIT
```

FORMAT 13 PROCESSING: NCSS EXECUTIVE = M99EXA

FILENAME: M99EXA

FILETYPE: EXEC

```
    &TYPE OFF
    &COMMENT G.P. END TRANS PROCESSING MSG
    &COMMENT SINGLE ARG IS : (1) TRANSACTION NUMBER
    &SPACE
    &PRINT TRANSACTION &1 PROCESSING COMPLETED
STATE FILE FT04F001 P
    &IF &INDEX0 NE 0 &EXIT
ERASE FILE FT04F001 P
    &EXIT
```

FORMAT I3 PROCESSING: RAMIS RUN PROCEDURE - M13RP

FILENAME: M13RP

FILETYPE: RUNPROC

```
DELETE
FILE RAMEXEC
1=OUT=M13RP,$
END
CATALOG M13RP
&& * DELETE AN INSTRUCTOR ID
&& * BOTH COURSE AND INSTRUCTOR FILES HANDLED HEREIN
&& * ENTRY ARGS ARE : (1) TRANS NUMBER, (2) FIRST 6 DIGITS OF THE
&& *       INSTR ID TO BE DELETED, (3) LAST 3 DIGITS OF THE ID,
&& *       (4) INSTR FIELD NAME, E.G., F1, (5) SPIN FILE LRECL,
&& *       (6) INSTR FIELD NUMBER IN COURSE FILE DESCRIPTION //
&& *
&& ARGS CONCAT=.
&& PRINT SPACE
&& PRINT NOW PROCESSING TRANSACTION NUMBER &1
&& PRINT INSTRUCTOR ID TO BE DELETED IS &2.&3
DELETE
FILE INST
ID=OUT=&2.&3,$
END
&& RUN
&& &ER = 'NO ERRORS' ;
&& IF &COND EQ 0 THEN GOTO S1 ;
&& &ER = 'ERRORS' ;
&&S1
&& PRINT SPACE
&& PRINT &ER DETECTED IN STEP1; RAMIS INDICATORS ARE :
&& PRINT &COND , &NOKEYS , &REJECTS , &RECORDS ,
&& PRINT &INPUTS , &TRANS , &LINES
&& *
TABLEF
FILE COURSE
PRINT COP AND &4 AND SAVE
IF &4 IS &2.&3
END
&& RUN
&& &ER = 'NO ERRORS' ;
&& IF &COND NE 0 THEN GOTO S2 ;
&& IF &RECORDS NE 0 THEN GOTO S2A ;
&&S2
&& &ER = 'ERRORS' ;
&&S2A
&& PRINT SPACE
&& PRINT &ER DETECTED IN STEP2; RAMIS INDICATORS ARE :
&& PRINT &COND , &NOKEYS , &REJECTS , &RECORDS ,
&& PRINT &INPUTS , &TRANS , &LINES
&& *
```

FORMAT 13 PROCESSING: RAMIS RUN PROCEDURE = M13RP (CONTINUED

FILENAME: M13RP

FILETYPE: RUNPROC

```
&& IF &COND NE 0 THEN GOTO &BROKE ;
&& IF &RECORDS NE 0 THEN GOTO S3X.;
&& *
&&BROKE
&& PRINT SPACE
&& PRINT DUE TO PREVIOUS ERRORS NOTED IN STEP2, ABOVE, STEP3
&& PRINT PROCESSING OF TRANSACTION NUMBER &1 IS BEING
&& PRINT BYPASSED...
&& EXIT
&& *
&&S3X
DELETE
READ FILE FT04F001
RECORD &5
FORM 4,9
ORDER 1,&6
SET &6
FILE COURSE
END
&& RUN
&& &ER = 'NO ERRORS' ;
&& IF &COND NE 0 THEN GOTO S3 ;
&& IF &CHANGES NE 0 THEN GOTO S3A ;
&&S3
&& &ER = 'ERRORS' ;
&&S3A
&& PRINT SPACE
&& PRINT &ER DETECTED IN STEP3; RAMIS INDICATORS ARE :
&& PRINT &COND , &NOKEYS , &REJECTS , &RECORDS ,
&& PRINT &INPUTS , &TRANS , &LINES
END CATALOG
```

TRANSACTION FORMAT : 14

FUNCTION : Delete a Building/Room Segment from the Facilities File

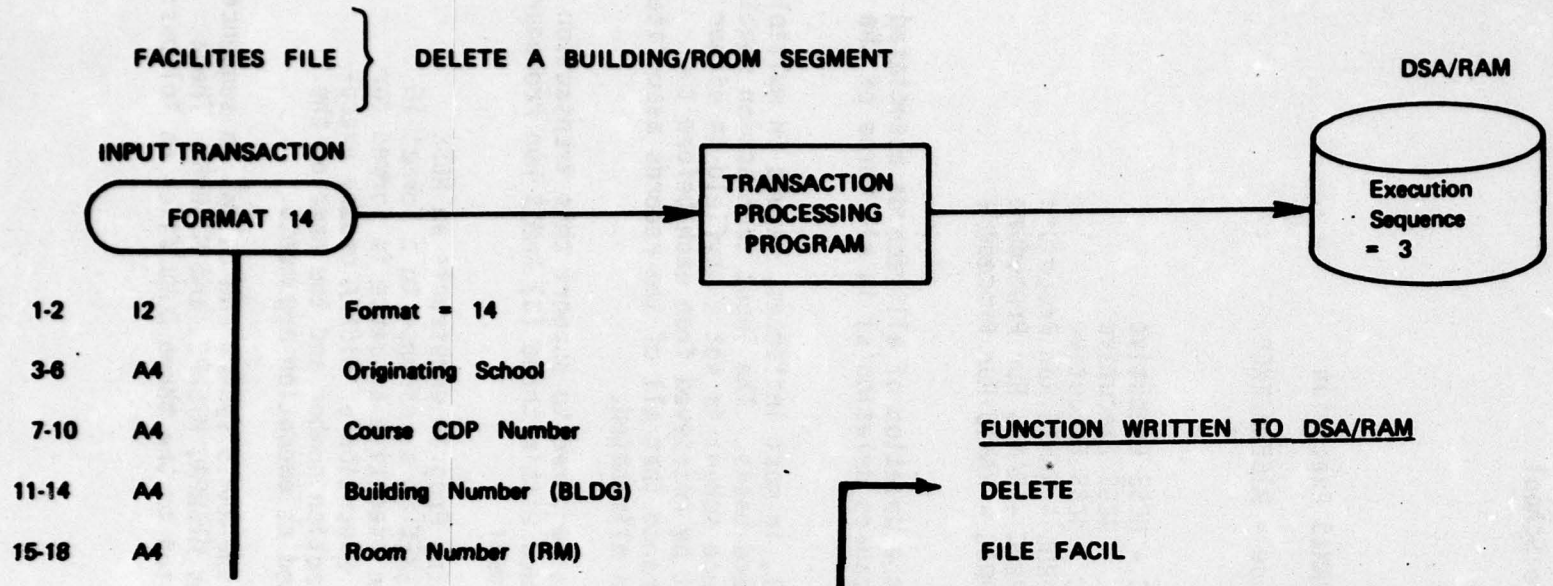
DATA BASE(S) MODIFIED : Facilities File

TRANSACTION TYPE : General Type A
Single Step RAMIS Execution

TRANSACTION FILE : Filename/Filetype = DSA/RAM

EXECUTION SEQUENCE : 3

This transaction commands the deletion of an entire building/room (BLDG/RM) lower level segment from the DOTS Facilities file. The high level portion of the Facilities record (key field = course CDP number) remains unchanged after the deletion.



1-2	I2	Format = 14
3-6	A4	Originating School
7-10	A4	Course CDP Number
11-14	A4	Building Number (BLDG)
15-18	A4	Room Number (RM)

NOTES

(1) One DELETE function for each Format 14 input transaction

(2) Items in lower case have been transcribed from the input transaction

FUNCTION WRITTEN TO DSA/RAM

DELETE

FILE FACIL

CDP=cdpnumber,RM=rmnumber,BLDG=OUT= bldgnumber.$

END

FORMAT 14 INPUT TRANSACTION PROCESSING

TRANSACTION FORMAT : 15

FUNCTION : Delete an Entire School

DATA BASE(S) MODIFIED : Facilities File
Instructor File
Course File

TRANSACTION TYPE : General Type E
Multiple Step RAMIS Execution

TRANSACTION FILE : Filename/Filetype = M15EXA/EXEC

EXECUTION SEQUENCE : 6

ADDITIONAL SUPPORT : (1) M15EXB/EXEC - NCSS Executive
(2) M15EXC/EXEC - NCSS Executive
(3) M99EXA/EXEC - NCSS Executive
(4) M15RPA/RUNPROC - RAMIS Run Procedure
(5) M15RPC/RUNPROC - RAMIS Run Procedure
(6) M15RPB/RUNPROC - RAMIS Run Procedure

A Format 15 input transaction commands the deletion of all records associated with a given school. This transaction causes deletion(s) in all three of the DOTS/TRAPAC files.

Processing of this transaction type will, in most instances, result in multiple record deletions in each of the three data bases. The input transaction specifies only the school name to be deleted. Since school is not a keyfield in either of the three files, additional data must be retrieved from each before the deletion can be commanded with the assurance that all of the records associated with the deleted school have indeed been eliminated.

Six RAMIS maintenance function steps are required to support this transaction type. All of these functions are included within three (3) RAMIS Run Procedures. The deletion operation proceeds as follows:

(1) The Input Transaction Processing Program constructs an NCSS executive file called M15EXA/EXEC as a response to a Format 15 input transaction. A separate executive sequence is formed for each unique Format 15 input. Execution of M15EXA causes execution M15EXB - the input transaction number and the name of the school to be deleted are passed as execution arguments.

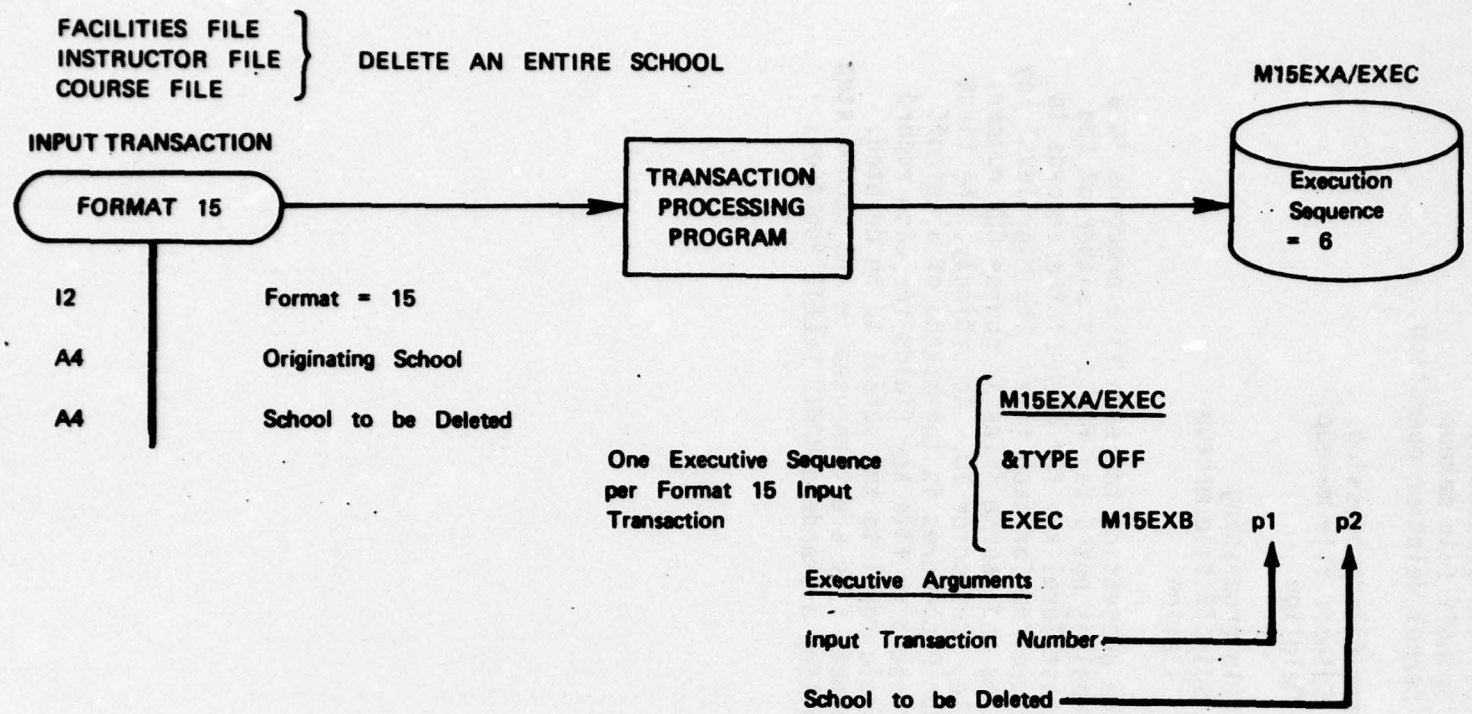
(2) Executive M15EXB initializes the RAMIS system and calls, in sequence, the three RAMIS Run Procedures M15RPA, M15RPC, and M15RPB. These three Run Procedures are related to the three DOTS files as follows:

M15RPA - Facilities file processing
STEP1 = Spinoff file makeup
STEP2 = School deletion operation

M15RPC - Instructor file processing
STEP3 = Spinoff file makeup
STEP4 - Deletion

M15RPB - Course file processing
STEP5 = Spinoff file makeup
STEP6 = Deletion

(3) As indicated above, the processing for each file proceeds in a parallel fashion. School is not a key field in either of the three files. The highest level key for each of the records to be deleted must be determined for each file. The high level key field for the Facilities and Course files is course CDP number; for the Instructor file, instructor ID. Accordingly, the first step in each of the Run Procedures is the makeup of a spinoff file containing the respective file key fields for each record containing a school field equal to the school to be deleted. The records of the spinoff file are then used in the second step as transaction controlling a standard RAMIS DELETE function.



FORMAT 15 INPUT TRANSACTION PROCESSING

FORMAT 15 PROCESSING: NCSS EXECUTIVE = M15EXB

FILENAME: M15EXB

FILETYPE: EXEC

```
    &TYPE OFF
    &COMMENT DELETE A SCHOOL. 2ND-LEVEL EXEC.
    &COMMENT ENTRY ARGS ARE :
    &COMMENT (1) TRANSACTION NUMBER / (2) SCHOOL CODE //
STATE FILE FT04F001 P
    &IF &INDEX0 NE 0 &GOTO -S1
ERASE FILE FT04F001 P
-S1 FILEDEF 4 DSK FILE FT04F001 RE F LR 8 BL 8
    &STACK RT
    &STACK EXEC M15RPA &1 , &2
    &STACK FIN
ATTACH RAMIS ALL BR
EXEC M15EXC
    &STACK EXEC M15RPC &1 , &2
    &STACK FIN
EXEC M15EXC
FILEDEF 4 CLEAR
STATE FILE FT04F001 P
    &IF &INDEX0 NE 0 &GOTO -S2
ERASE FILE FT04F001 P
-S2 FILEDEF 4 DSK FILE FT04F001 RE F LR 13 BL 13
    &STACK EXEC M15RPB &1 , &2
    &STACK FIN
EXEC M15EXC
DETACH RAMIS BR
    &STACK RT
EXEC M99EXA &1
    &EXIT
```

FORMAT 15 PROCESSING: NCSS EXECUTIVE = M15EXC

FILENAME: M15EXC

FILETYPE: EXEC

```
    &TYPE OFF
    &COMMENT INTERNAL EXEC FOR DELETE SCHOOL.
    &COMMENT THIS EXEC MERELY PRINTS THE MSG/KILL FILE.
    &COMMENT NO ENTRY ARGS.
FILEDEF 6 DSK MSG KILL
EXEC RAMIS
    &STACK RT
FILEDEF 6 CONO
    &STACK KT
    &STACK L. /SPACE/
    &STACK C /SPACE//
    &STACK DOWN.
    &STACK P 4
    &STACK G
EDIT MSG KILL
    &EXIT
```

FORMAT 15 PROCESSING: NCSS EXECUTIVE = M99EXA

FILENAME: M99EXA

FILETYPE: EXEC

```
    &TYPE OFF
    &COMMENT G.P. END TRANS PROCESSING MSG
    &COMMENT SINGLE ARG IS : (1) TRANSACTION NUMBER
    &SPACE
    &PRINT TRANSACTION &1 PROCESSING COMPLETED
STATE FILE FT04F001 P
    &IF &INDEXQ NE 0 &EXIT
ERASE FILE FT04F001 P
    &EXIT
```

FORMAT 15 PROCESSING: RAMIS RUN PROCEDURE = M15RPA

FILENAME: M15RPA

FILETYPE: RUNPROC

```
DELETE
FILE RAMEXEC
1=OUT=M15RPA,$
END
CATALOG M15RPA
&& * DELETE AN ENTIRE SCHOOL; SEE ALSO M15RPB, M15RPC
&& * FACILITIES FILE PROCESSING ONLY
&& * ENTRY ARGS : (1) TRANS NUMBER, (2) SCHOOL TO BE DELETED //
&& *
&& ARGS CONCAT=,
&& &STEP = 'STEP1' ;
TABLEF
FILE FACIL
PRINT CDP AND SCH AND SAVE
IF SCH IS &2
END
&& RUN
&& &ER = 'NO ERRORS' ;
&& IF &COND NE 0 THEN GOTO BUST ;
&& IF &RECORDS EQ 0 THEN GOTO BUST ;
&& *
&& &STEP = 'STEP2' ;
DELETE
READ FILE FT04F001
RECORD 8
FORM 4,4
ORDER 1,2
SET 1
FILE FACIL
END
&& RUN
&& IF &COND NE 0 THEN GOTO BUST ;
&& IF &CHANGES NE 0 THEN GOTO S3 ;
&& *
&&BUST
&& &ER = 'ERRORS WERE' ;
&&S3
&& PRINT SPACE
&& PRINT TRANSACTION NO. &1 - FACILITIES FILE DELETE OF SCHOOL = &2
&& PRINT &ER DETECTED IN &STEP ; RAMIS INDICATORS ARE :
&& PRINT &COND , &NOKEYS , &REJECTS , &RECORDS
&& PRINT &CHANGES , &INPUTS , &TRANS , &LINES
END CATALOG
```

FORMAT 15 PROCESSING: RAMIS RUN PROCEDURE = M15RPB

FILENAME: M15RPB

FILETYPE: RUNPROC

```
DELETE
FILE RAMEXEC
1=OUT=M15RPH,$
END
CATALOG M15RPB
&& * DELETE AN ENTIRE SCHOOL; SEE ALSO M15RPA, M15RPC
&& * INSTRUCTOR FILE PROCESSING ONLY
&& * ENTRY ARGS ARE : (1) TRANS NUMBER, (2) SCHOOL TO BE DELETED //
&& *
&& ARGS CONCAT=,
&& &STEP = 'STEP5' ;
TABLEF
FILE INST
PRINT ID AND SCH AND SAVE
IF SCH IS &2
END
&& RUN
&& &ER = 'NO ERRORS' ;
&& IF &COND NE 0 THEN GOTO BUST ;
&& IF &RECORDS EQ 0 THEN GOTO BUST ;
&& *
&& &STEP = 'STEP6' ;
DELETE
READ FILE FT04F001
RECORD 13
FORM 9,4
ORDER 1,5
SET 1
FILE INST
END
&& RUN
&& IF &COND NE 0 THEN GOTO BUST ;
&& IF &CHANGES NE 0 THEN GOTO S3 ;
&& *
&&BUST
&& &ER = 'ERRORS WERE' ;
&&S3
&& PRINT SPACE
&& PRINT TRANSACTION NO. &1 - INSTRUCTOR FILE DELETE OF SCHOOL = &2
&& PRINT &ER DETECTED IN &STEP ; RAMIS INDICATORS ARE :
&& PRINT &COND , &NOKEYS , &REJECTS , &RECORDS
&& PRINT &CHANGES , &INPUTS , &TRANS , &LINES
END CATALOG
```

FORMAT 15 PROCESSING: RAMIS RUN PROCEDURE = M15RPC

FILENAME: M15RPC

FILETYPE: RUNPROC

```
DELETE
FILE RAMEXEC
1=OUT=M15RPC,$
END
CATALOG M15RPC
&& * DELETE AN ENTIRE SCHOOL; SEE ALSO M15RPA, M15RPB
&& * COURSE FILE PROCESSING ONLY
&& * ENTRY ARGS ARE : (1) TRANS NUMBER, (2) SCHOOL TO BE DELETED //
&& *
&& ARGS CONCAT=.
&& &STEP = 'STEP3' ;
TABLEF
FILE COURSE
PRINT CDP AND SCH AND SAVE
IF SCH IS &2
END
&& RUN
&& &ER = 'NO ERRORS' ;
&& IF &COND NE 0 THEN GOTO BUST ;
&& IF &RECORDS EQ 0 THEN GOTO BUST ;
&& *
&& &STEP = 'STEP4' ;
DELETE
READ FILE FT04F001
RECORD 8
FORM 4,4
ORDER 1,2
SET 1
FILE COURSE
END
&& RUN
&& IF &COND NE 0 THEN GOTO BUST ;
&& IF &CHANGES NE 0 THEN GOTO S3 ;
&& *
&&BUST
&& &ER = 'ERRORS WERE' ;
&&S3
&& PRINT SPACE
&& PRINT TRANSACTION NO. &1 - COURSE FILE DELETE OF SCHOOL = &2
&& PRINT &ER DETECTED IN &STEP ; RAMIS INDICATORS ARE :
&& PRINT &COND , &NOKEYS , &REJECTS , &RECORDS
&& PRINT &CHANGES , &INPUTS , &TRANS , &LINES
END CATALOG
```

TRANSACTION FORMAT : 16

FUNCTION : Change the CDP Number of a Course

DATA BASE(S) MODIFIED : Facilities File
Course File

TRANSACTION TYPE : General Type E
Multiple Step RAMIS Execution

TRANSACTION FILE : Filename/Filetype = M16EXA/EXEC

EXECUTION SEQUENCE : 7

ADDITIONAL SUPPORT : (1) M16EXB/EXEC - NCSS Executive
(2) M15EXC/EXEC - NCSS Executive
(3) M99EXA/EXEC - NCSS Executive
(4) M16RPA/RUNPROC - RAMIS Run Procedure
(5) M16RPB/RUNPROC - RAMIS Run Procedure
(6) M16RPC/RUNPROC - RAMIS Run Procedure

This transaction type causes the change of the CDP number of an existing course. Course CDP number does not appear in the DOTS Instructor file. The Facilities and Course files have CDP number as their highest level key field.

Since the Facilities and Course files are maintained in sorted order by CDP number, a modification of this number via the normal RAMIS CHANGE function cannot be accomplished. To do so would mean placing the modified record back into the file out of sort sequence. The proper procedure for changing a key sorted field is to delete the old record (in its entirety) and then re-enter the same data with the new key field.

To effect a CDP change, the user must specify only the old CDP (to be changed) and the new CDP value.

The Input Transaction Processing Program constructs an NCSS executive file named M16EXA/EXEC as the result of a Format 16 input transaction. A separate executive sequence will be formed for each individual Format 16 input encountered. Each M16EXA executive sequence will cause the execution of executive M16EXB; the transaction number, old CDP number, and new CDP number will be passed as arguments between executives.

The execution sequence associated with Format 16 transaction processing is as follows:

M16EXA - Invokes execution of executive M16EXB.

M16EXB - Initializes RAMIS system.

- Invokes RAMIS Run Procedure M16RPA (Facilities file processing):

STEP1 - Dumps the record bearing the old CDP as key (all segments are dumped) to a utility dump file.

STEP2 - Via a standard RAMIS DELETE function, the old CDP record and all associated lower level segments are deleted.

STEP3 - Using the record segments dumped in STEP1, above, as input transactions, a RAMIS INPUT function is used to reload the old record data with the new CDP number key.

- Invokes RAMIS Run Procedure M16RPB (Course file processing, first stage):

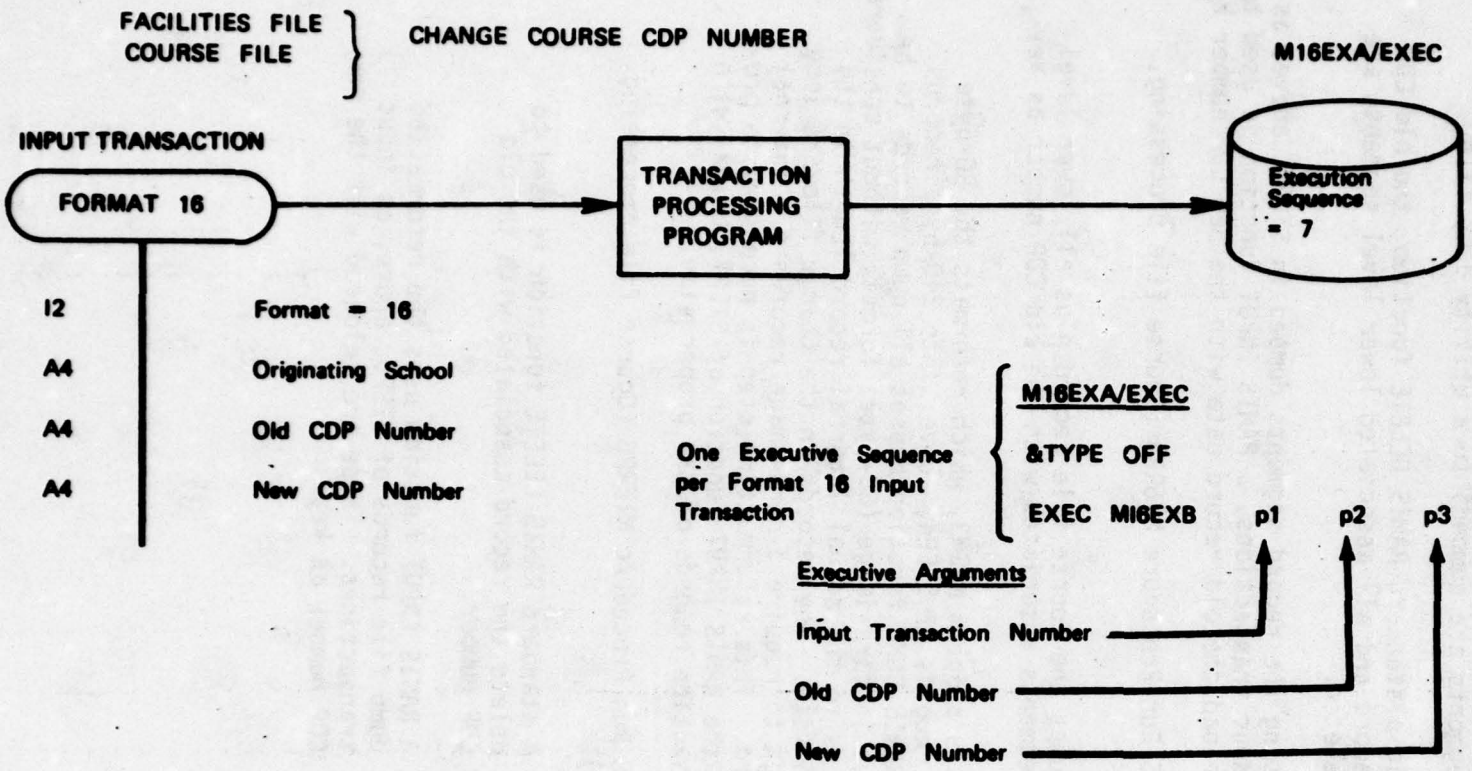
STEP4 - Dumps the Course file record plus all lower level segments associated with the old CDP number as key.

- STEP5 - Invokes program MPGM3, which reformats the 80-byte dump records of STEP4, above, into 240-byte records. The RAMIS DUMP function causes all dump records to be output in card-image (80-byte) format, without consideration as to the actual internal record length in the RAMIS file. The records in the Course file are long enough to require 3 card-image records per internal record. This reformatting step is necessary to provide the RAMIS INPUT function of STEP7, below, with transaction records of the proper size.

- Invokes RAMIS Run Procedure M16RPC (Course file processing, second stage):

STEP6 - A standard RAMIS DELETE function is used to delete the record associated with the old CDP number.

STEP7 - A RAMIS INPUT function uses the reformatted dump file records of STEP5, above, as input transactions. These are reloaded with the new CDP number as key.



FORMAT 16 INPUT TRANSACTION PROCESSING

FORMAT 16 PROCESSING: NCSS EXECUTIVE = M16EXB

FILENAME: M16EXB

FILETYPE: EXEC

```
    &TYPE OFF
    &COMMENT CHANGE-A-CDP•2ND-LEVEL EXEC.
    &COMMENT ENTRY ARGS ARE :
    &COMMENT (1) TRANS NUMBER / (2) OLD CDP /
    &COMMENT (3) NEW CDP //
STATE FACIL MDUMP P
    &IF &INDEX0 NE 0 &GOTO -S1
ERASE FACIL MDUMP P
-S1 FILEDEF RAMUTIL1 DSK FACIL MDUMP RE F LR 80 BL 80
    &STACK RT
    &STACK EXEC M16RPA &1 • &2 • &3
    &STACK FIN
ATTACH RAMIS ALL BR
EXEC M15EXC
FILEDEF RAMUTIL1 CLEAR
STATE COURSE MDUMP P
    &IF &INDEX0 NE 0 &GOTO -S2
ERASE COURSE MDUMP P
-S2 FILEDEF RAMUTIL1 DSK COURSE MDUMP RE F LR 80 BL 80
    &STACK EXEC M16RPB &1 • &2
    &STACK FIN
EXEC M15EXC
STATE COURSE MDUMP P
    &IF &INDEX0 NE 0 &GOTO -SOUT
STATE COURSE MDUMP2 P
    &IF &INDEX0 NE 0 &GOTO -S3
ERASE COURSE MDUMP2 P
-S3 FILEDEF RAMUTIL1 CLEAR
FILEDEF DUMP DSK COURSE MDUMP RE F LR 80 BL 80
FILEDEF DUMP2 DSK COURSE MDUMP2 RE F LR 240 BL 240
    &SPACE
LOAD MPGM3 (CLEAR LIBE) PLILIB
START IHECMS 3 (BRIEF)
    &SPACE
    &IF &INDEX0 EQ 0 &GOTO -S4
    &SPACE
    &PRINT REFORMAT (STEP5) OF COURSE FILE DUMP FAILED•
    &PRINT RETURN INDEX0 WAS &INDEX0
-SOUT &SPACE
    &PRINT PROCESSING OF TRANSACTION &1 TERMINATING ABNORMALLY
    &EXIT
-S4 &STACK EXEC M16RPC &1 • &2 • &3
    &STACK FIN
EXEC M15EXC
DETACH RAMIS BR
    &STACK RT
    &SPACE
    &PRINT TRANSACTION &1 PROCESSING COMPLETED
    &EXIT
```

FORMAT 16 PROCESSING: NCSS EXECUTIVE = M15EXC

FILENAME: M15EXC

FILETYPE: EXEC

```
    &TYPE OFF
    &COMMENT INTERNAL EXEC FOR DELETE SCHOOL.
    &COMMENT THIS EXEC MERELY PRINTS THE MSG/KILL FILE.
    &COMMENT NO ENTRY ARGS.
FILEDEF 6 DSK MSG KILL
EXEC RAMIS
    &STACK RT
FILEDEF 6 CONO
    &STACK KT
    &STACK L. /SPACE/
    &STACK C /SPACE//
    &STACK DOWN.
    &STACK P 4
    &STACK G
EDIT MSG KILL
    &EXIT
```

FORMAT 16 PROCESSING: NCSS EXECUTIVE = M99EXA

FILENAME: M99EXA

FILETYPE: EXEC

```
    &TYPE OFF
    &COMMENT G.P. END TRANS PROCESSING MSG
    &COMMENT SINGLE ARG IS : (1) TRANSACTION NUMBER
    &SPACE
    &PRINT TRANSACTION &1 PROCESSING COMPLETED
STATE FILE FT04F001 P
    &IF &INDEX0 NE 0 &EXIT
ERASE FILE FT04F001 P
    &EXIT
```

FORMAT 16 PROCESSING: RAMIS RUN PROCEDURE = M16RPA

FILENAME: M16RPA

FILETYPE: RUNPROC

```
DELETE
FILE RAMEXEC
1=OUT=M16RPA,$
END
CATALOG M16RPA
&& * CHANGE CDP PROCESSING.
&& * ENTRY ARGS ARE : (1) TRANS NUMBER ;
&& * (2) OLD CDP NUMBER ; (3) NEW CDP NUMBER //
DUMP
FILE FACIL
PRINT AVHRS AND SAVE
IF CDP IS &2
END
&& RUN
&& &ER = 'NO ERRORS' ;
&& &STEP = 'STEP1' ;
&& IF &COND NE 0 THEN GOTO BUST ;
&& IF &RECORDS EQ 0 THEN GOTO BUST ;
&& &STEP = 'STEP2' ;
DELETE
FILE FACIL
CDP=OUT=&2,$
END
&& RUN
&& IF &COND NE 0 THEN GOTO BUST ;
&& IF &CHANGES EQ 0 THEN GOTO BUST ;
&& &STEP = 'STEP3' ;
INPUT
READ FACIL MDUMP
RECORD 80
SET 1
FORM X4,4,5,3,3,1,6,5
ORDER 2,3,4,5,6,7,8
INSERT CDP = &3
FILE FACIL
END
&& RUN
&& IF &COND NE 0 THEN GOTO BUST ;
&& IF &INPUTS NE 0 THEN GOTO S3 ;
&&BUST
&& &ER = 'ERRORS WERE' ;
&&S3
&& PRINT SPACE
&& PRINT TRANSACTION NUMBER &1 - FACILITIES FILE PROCESSING
&& PRINT &ER DETECTED IN &STEP ; RAMIS INDICATORS ARE :
&& PRINT &COND , &NOKEYS , &REJECTS , &RECORDS
&& PRINT &CHANGES , &INPUTS , &TRANS , &LINES
END CATALOG
```

FORMAT 16 PROCESSING: RAMIS RUN PROCEDURE = M16RPB

FILENAME: M16RPB

FILETYPE: RUNPROC

```
DELETE
FILE RAMEXEC
1=OUT=M16RPB,$
END
CATALOG M16RPB
DUMP
FILE COURSE
PRINT F3 AND SAVE
IF CDP IS &2
END
&& RUN
&& &ER = 'NO ERRORS' ;
&& IF &COND NE 0 THEN GOTO BUST ;
&& IF &RECORDS NE 0 THEN GOTO S1 ;
&&BUST
&& &ER = 'ERRORS WERE' ;
&&S1
&& PRINT SPACE
&& PRINT TRANSACTION NUMBER &1 - COURSE FILE PROCESSING
&& PRINT &ER DETECTED IN STEP4 ; RAMIS INDICATORS ARE :
&& PRINT &COND , &NOKEYS , &REJECTS , &RECORDS ,
&& PRINT &CHANGES , &INPUTS , &TRANS , &LINES
END CATALOG
```

FORMAT 16 PROCESSING: RAMIS RUN PROCEDURE = M16RPC

FILENAME: M16RPC

FILETYPE: RUNPROC

```
DELETE
FILE RAMEXEC
1=OUT=M16RPC,$
END
CATALOG M16RPC
&& &STEP = 'STEP6' ;
&& &ER = 'NO ERRORS' ;
DELETE
FILE COURSE
CDP=OUT=&2,$
END
&& RUN
&& IF &COND NE 0 THEN GOTO BUST ;
&& IF &CHANGES EQ 0 THEN GOTO BUST ;
&& &STEP = 'STEP7' ;
INPUT
READ COURSE MDUMP2
RECORD 240
SET 1
FORM X4,4,1,10,16,3,4,3,3,5,5,2,5,5,5,5
FORM 4,4,3,4,3,3,5,5,3,3,4,3,3,5,5,3
FORM 4,5,2,2,4,3,3,2,6,5,4,5,10
FORM 6,6,6,3,3,3,4,9,4,1
ORDER 2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
ORDER 17,18,19,20,21,22,23,24,25,26,27,28,29
ORDER 30,31,32,33,34,35,36,37,38,39,40,41,42
ORDER 43,44,45,46,47,48,49,50,51,52,53,54,55
INSERT CDP = &3
FILE COURSE
END
&& RUN
&& IF &COND NE 0 THEN GOTO BUST ;
&& IF &INPUTS NE 0 THEN GOTO S2 ;
&&BUST
&& &ER = 'ERRORS WERE' ;
&&S2
&& PRINT SPACE
&& PRINT TRANSACTION NUMBER &1 - COURSE FILE PROCESSING
&& PRINT &ER DETECTED IN &STEP - RAMIS INDICATORS ARE :
&& PRINT &COND , &NOKEYS , &REJECTS , &RECORDS ,
&& PRINT &CHANGES , &INPUTS , &TRANS , &LINES
END CATALOG
```

TRANSACTION FORMAT : 17

FUNCTION : Change a Student/Instructor Ratio Segment

DATA BASE(S) MODIFIED : Course File

TRANSACTION TYPE : General Type E
Multiple Step RAMIS Execution

TRANSACTION FILE : Filename/Filetype = M17EXA/EXEC

EXECUTION SEQUENCE : 8

ADDITIONAL SUPPORT :
(1) M17EXB/EXEC - NCSS Executive
(2) M15EXC/EXEC - NCSS Executive
(3) M99EXA/EXEC - NCSS Executive
(4) M17RP/RUNPROC - RAMIS Run Procedure

Student/instructor ratio (SIR) is the key field for the second-level SIR segment of the DOTS Course File. A Format 17 transaction is used to effect a change in the SIR field, without changing any of the other fields of the SIR segment.

Within each Course file record, the SIR segments are maintained in sorted order by SIR field. The mnemonic used for the SIR field is 'F1.' If a RAMIS CHANGE function were used to change the value of an F1 field, the changed record may be subsequently out of sort sequence. The CHANGE function merely changes field values as commanded - no reordering is accomplished.

The normal method of accomplishing such key field maintenance is to delete the old segment (in its entirety), and then to re-enter the removed segment in its entirety) with the new keyfield value. To avoid making the DOTS user enter all of the additional segment field values when only a keyfield change was of interest, the segment removal and replacement is accomplished in a multi-step RAMIS Run Procedure called M17RP.

Transaction Format 17 maintenance ensues as follows:

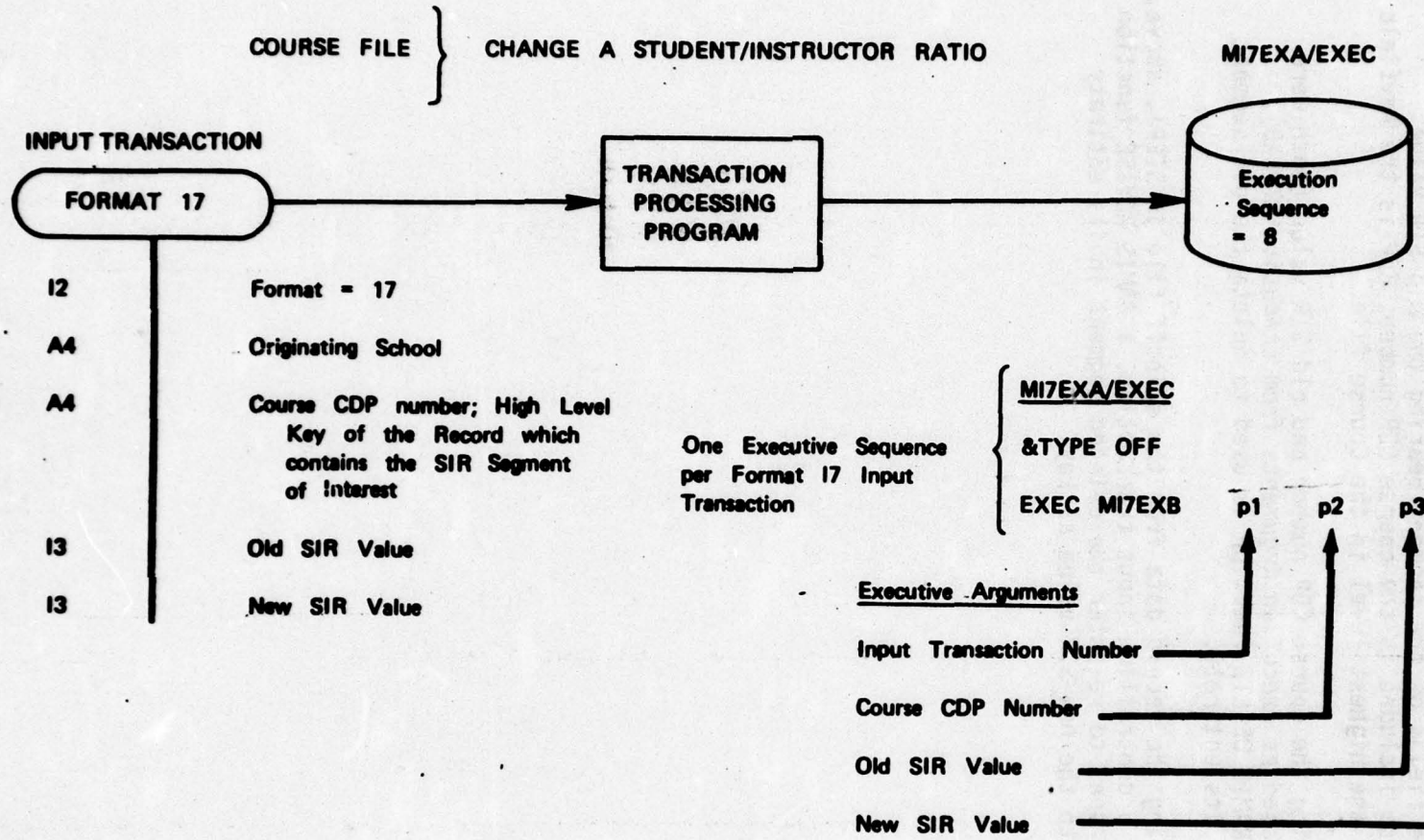
(1) The Transaction Processing Program produces as output one M17EXA Executive sequence per Format 17 input transaction. Each M17EXA invokes a separate execution of executive M17EXB, passing the transaction sequence number, the course CDP number, the old SIR value, and the new SIR value as execution arguments.

(2) After initialization of the RAMIS system, executive M17EXB invokes RAMIS Run Procedure M17RP. Within M17RP, the following is accomplished:

STEP1 - A RAMIS spinoff file is constructed containing all of the fields of the segment bearing the old SIR value. Also included is the course CDP number; CDP is the keyfield at the highest level in the Course file.

STEP2 - Using the course CDP number and old SIR value (both were passed as execution arguments from executive M17EXB), a RAMIS DELETE function is used to delete the old segment in its entirety.

STEP3 - Using the record data from the spinoff file in STEP1, above, as a controlling input transaction, a RAMIS REVISE function is used to re-enter the deleted segment in its entirety with the new SIR value as key.



FORMAT 17 INPUT TRANSACTION PROCESSING

FORMAT 17 PROCESSING: NCSS EXECUTIVE = M17EXB

FILENAME: M17EXB

FILETYPE: EXEC

```
    &TYPE OFF
    &COMMENT CHANGE A STUD/INSTR (SIR) SEGMENT.
    &COMMENT 2ND-LEVEL EXEC.
    &COMMENT ENTRY ARGS ARE :
    &COMMENT (1) TRANS NUMBER / (2) CDF NUMBER /
    &COMMENT (3) OLD SIR / (4) NEW SIR //
STATE FILE FT04F001 P
    &IF &INDEX0 NE 0 &GOTO -S1
ERASE FILE FT04F001 P
-S1 FILEDEF 4 DSK FILE FT04F001 RE F LR 18 RL 18
    &STACK RT
    &STACK EXEC M17RP &1 . &2 . &3 .
    &STACK &4
    &STACK FIN
ATTACH RAMIS ALL BR
EXEC M15EXC
DETACH RAMIS BR
    &STACK RT
EXEC M99EXA &1
    &EXIT
```

FORMAT 17 PROCESSING: NCSS EXECUTIVE = M15EXC

FILENAME: M15EXC

FILETYPE: EXEC

```
    &TYPE OFF
    &COMMENT INTERNAL EXEC FOR DELETE SCHOOL.
    &COMMENT THIS EXEC MERELY PRINTS THE MSG/KILL FILE.
    &COMMENT NO ENTRY ARGS.
FILEDEF 6 DSK MSG KILL
EXEC RAMIS
    &STACK RT
FILEDEF 6 CONO
    &STACK RT
    &STACK L. /SPACE/
    &STACK C /SPACE//
    &STACK DOWN.
    &STACK P 4
    &STACK 6
EDIT MSG KILL
    &EXIT
```

FORMAT 17 PROCESSING: NCSS EXECUTIVE = M99EXA

FILENAME: M99EXA

FILETYPE: EXEC

```
    &TYPE OFF
    &COMMENT G.P. END TRANS PROCESSING MSG
    &COMMENT SINGLE ARG IS : (1) TRANSACTION NUMBER
    &SPACE
    &PRINT TRANSACTION &1 PROCESSING COMPLETED
STATE FILE FT04F001 P
    &IF &INDEX0 NE 0 &EXIT
ERASE FILE FT04F001 P
    &EXIT
```

FORMAT 17 PROCESSING: RAMIS RUN PROCEDURE = M17RP

FILENAME: M17RP

FILETYPE: RUNPROC

```
DELETE
FILE RAMEXEC
1=OUT=M17RP,$
END
CATALOG M17RP
&& &ER = 'NO ERRORS' ;
&& &STEP = 'STEP1' ;
TABLE
FILE COURSE
PRINT CDP AND F1 AND F2 AND F3 AND SAVE
IF CDP IS &2
IF F1 IS &3
END
&& RUN
&& IF &COND NE 0 THEN GOTO BUST ;
&& IF &RECORDS EQ 0 THEN GOTO BUST ;
&& &STEP = 'STEP2' ;
DELETE
FILE COURSE
CDP=&2,F1=OUT=&3,$
END
&& RUN
&& IF &COND NE 0 THEN GOTO BUST ;
&& IF &CHANGES EQ 0 THEN GOTO BUST ;
&& &STEP = 'STEP3' ;
REVISE
READ FILE FT04F001
RECORD 18
FORM 4,X9,4,1
ORDER 1,54,55
INSERT F1 = &4
KEY 1,53
INCLUDE 53
FILE COURSE
END
&& RUN
&& IF &COND NE 0 THEN GOTO BUST ;
&& IF &INPUTS NE 0 THEN GOTO S1 ;
&&BUST
&& &ER = 'ERRORS WERE' ;
&&S1
&& PRINT SPACE
&& PRINT TRANSACTION NUMBER &1 - COURSE FILE PROCESSING ONLY
&& PRINT &ER DETECTED IN &STEP ; RAMIS INDICATORS ARE :
&& PRINT &COND , &NOKEYS , &REJECTS , &RECORDS ,
&& PRINT &CHANGES , &INPUTS , &TRANS , &LINES
END CATALOG
```

TRANSACTION FORMAT : 18

FUNCTION : Change an Existing Instructor ID

DATA BASE(S) MODIFIED: Instructor File
Course File

TRANSACTION TYPE : General Type E
Multiple Step RAMIS Execution

TRANSACTION FILE : Filename/Filetype = M18EXA/EXEC

EXECUTION SEQUENCE : 9

ADDITIONAL SUPPORT : (1) M18EXB/EXEC - NCSS Executive
(2) M15EXC/EXEC - NCSS Executive
(3) M99EXA/EXEC - NCSS Executive
(4) M18RPA/RUNPROC - RAMIS Run Procedure
(5) M18RPB/RUNPROC - RAMIS Run Procedure
(6) M18RPC/RUNPROC - RAMIS Run Procedure

This transaction type forces the change of an existing instructor ID (Social Security Number). This field does not appear at all in the DOTS Facilities file. Within the Instructor file, ID is the keyfield at the highest level. In the Course file, ID is key to the second-level segment (course CDP number is the highest level key in the Course file).

Since instructor ID is key within both files, change of this key value is a two-step operation. First, the old record or record segment is deleted. Second, the old record or segment is reintroduced into the file with the new value of ID as key.

The Transaction Processing Program produces one M18EXA executive sequence for each Format 18 input transaction. Each M18EXA executive will cause the execution of executive M18EXB on a one-for-one basis. Passed as arguments between these executives are the transaction sequence number, the old instructor ID, and the new instructor ID. Processing at this point proceeds as follows:

M18EXB - This executive initializes the RAMIS system and invokes the RAMIS Run Procedure M18RPA. Within this Run Procedure:

STEP1 - A RAMIS DUMP function is used to create a dump file from the DOTS Instructor file. To be dumped is the entire instructor record keyed by the old ID field.

STEP2 - After control is returned to executive M18EXB, Program MPGM3 is invoked to reformat the dump file created in STEP1, above. A RAMIS dump file is

established in card-image (80 Byte) format, regardless of the length of the record internal to the RAMIS file being dumped. The internal length of the Instructor file record is such that two card-images per record are required. After reformatting, the dump file record length will be 160 Bytes.

If the reformatting operation of STEP2 is not successful, further Instructor file processing is bypassed. Control then passes to STEP5, below. A successful operation during STEP2 will cause control to be passed to RAMIS Run Procedure M18RPB.

STEP3 - The entire record keyed by the old instructor ID in the Instructor file is deleted via a standard RAMIS DELETE function.

STEP4 - A RAMIS INPUT function is used to reinsert the deleted record bearing the new instructor ID key. The contents of the reformatted dump file produced in STEP2, above, are used as the transaction controlling the INPUT function.

At this point, RAMIS Run Procedure M18RPC is invoked. This Run Procedure will accomplish the Course file processing required for instructor ID change. Note that instructor ID is key for the second-level segment of the Course file; Course CDP number is key at the highest level. The same instructor ID may appear associated with several distinct courses.

STEP5 - A RAMIS spinoff file is created. Each record contains the course CDP number and all second-level field values for each instance of the old instructor ID.

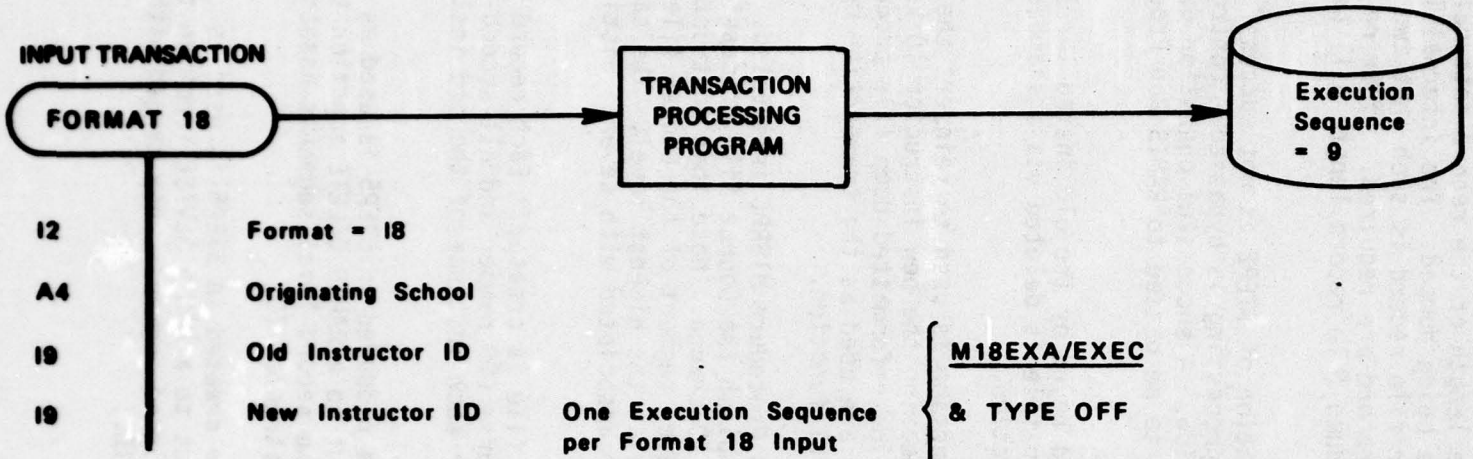
STEP6 - The spinoff file produced in STEP5 is used as transaction input to a RAMIS DELETE function to delete all of the second-level segments associated with the old instructor ID.

STEP7 - The spinoff file created in STEP5 is used as transaction input to a RAMIS REVISE function to re-enter the deleted segments, now tagged with the new instructor ID.

INSTRUCTOR FILE
COURSE FILE

CHANGE AN EXISTING INSTRUCTOR ID

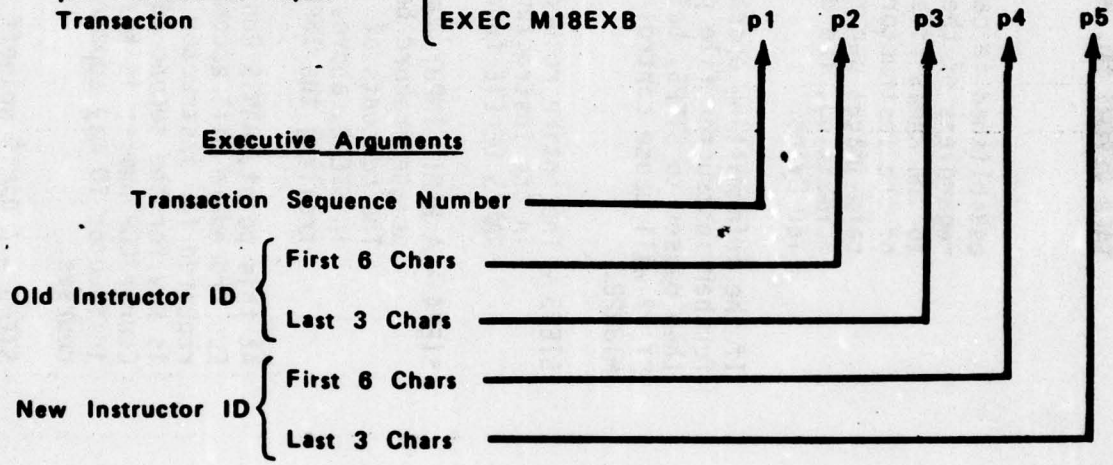
M18EXA/EXEC



1-2	I2	Format = 18
3-6	A4	Originating School
7-15	I9	Old Instructor ID
16-24	I9	New Instructor ID

One Execution Sequence per Format 18 Input Transaction

M18EXA/EXEC
& TYPE OFF
EXEC M18EXB



FORMAT 18 INPUT TRANSACTION PROCESSING

FORMAT 18 PROCESSING: NCSS EXECUTIVE = M18EXB

FILENAME: M18EXB

FILETYPE: EXEC

```
    &TYPE OFF
    &COMMENT CHANGE INSTRUCTOR ID. 2ND-LEVEL EXEC.
    &COMMENT ENTRY ARGS ARE :
    &COMMENT (1) TRANS NUMBER / (2) OLD ID. FIRST 6 CHARS /
    &COMMENT (3) OLD ID. LAST 3 CHAR / (4) NEW ID. FIRST 6 /
    &COMMENT (5) NEW ID. LAST 3 //
STATE INST MDUMP P
    &IF &INDEX0 NE 0 &GOTO -S1
ERASE INST MDUMP P
-S1 FILEDEF RAMUTIL1 DSK INST MDUMP RE F LR 80 BL 80
    &STACK EXEC M18RPA &1 . &2 . &3 .
    &STACK &4 . &5
    &STACK FIN
ATTACH RAMIS ALL BR
FILEDEF 6 DSK MSG KILL
EXEC RAMIS
    &STACK RT
FILEDEF 6 CONO
    &STACK KT
    &STACK L. /SPACE/
    &STACK C /SPACE//
    &STACK DOWN.
    &STACK P 2
    &STACK L. /SPACE/
    &STACK C /SPACE//
    &STACK DOWN.
    &STACK P 4
    &STACK Q
EDIT MSG KILL
FILEDEF RAMUTIL1 CLEAR
STATE INST MDUMP P
    &IF &INDEX0 EQ 0 &GOTO -S2
    &SPACE
    &PRINT INSTRUCTOR FILE ID DUMP FAILED -
-S1 &PRINT BYPASSING FURTHER INST FILE PROCESSING.
    &PRINT PROCEEDING WITH COURSE FILE PROCESSING.
    &GOTO -S6
-S2 STATE INST MDUMP2 P
    &IF &INDEX0 NE 0 &GOTO -S3
ERASE INST MDUMP2 P
-S3 FILEDEF DUMP DSK INST MDUMP RE F LR 80 BL 80
FILEDEF DUMP2 DSK INST MDUMP2 RE F LR 160 BL 160
    &SPACE
```

NCSS EXECUTIVE M18EXB (CONTINUED)

```
    LOAD MPGM3 (CLEAR LIBE) PLILIB
    START IHECMS 2 (BRIEF)
         &SPACE
         &IF &INDEX0 EQ 0 &GOTO -S4
         &PRINT REFORMAT (STEP2) OF INST FILE DUMP FAILED.
         &PRINT RETURN INDEX0 WAS &INDEX0 -
         &GOTO -S1A
    -S4 &STACK EXEC M18RPB &1 , &2 , &3 ,
         &STACK &4 , &5
         &STACK FIN
    EXEC M15EXC
    -S6 STATE FILE FT04F001 P

        &IF &INDEX0 NE 0 &GOTO -S5
    ERASE FILE FT04F001 P
    -S5 FILEDEF 4 DSK FILE FT04F001 RE F LR 18 BL 18
         &STACK RT
         &STACK EXEC M18RPC &1 , &2 , &3 ,
         &STACK &4 , &5
         &STACK FIN
    EXEC M15EXC
    DETACH RAMIS RR
         &STACK RT
    EXEC M99EXA &1
         &EXIT
```

FORMAT 18 PROCESSING: MCSS EXECUTIVE = M15EXC

FILENAME: M15EXC

FILETYPE: EXEC

```
     &TYPE OFF
     &COMMENT INTERNAL EXEC FOR DELETE SCHOOL.
     &COMMENT THIS EXEC MERELY PRINTS THE MSG/KILL FILE.
     &COMMENT NO ENTRY ARGS.
FILEDEF 6 DSK MSG KILL
EXEC RAMIS
     &STACK RT
FILEDEF 6 CONO
     &STACK RT
     &STACK L. /SPACE/
     &STACK C /SPACE//
     &STACK DOWN.
     &STACK P 4
     &STACK G
EDIT MSG KILL
     &EXIT
```

FORMAT 18 PROCESSING: NCSS EXECUTIVE = M99EXC

FILENAME: M99EXC

FILETYPE: EXEC

```
    &TYPE OFF
    &COMMENT G.P. END TRANS PROCESSING MSG
    &COMMENT SINGLE ARG IS : (1) TRANSACTION NUMBER
    &SPACE
    &PRINT TRANSACTION &1 PROCESSING COMPLETED
STATE FILE FT04F001 P
    &IF &INDEX0 NE 0 &EXIT
ERASE FILE FT04F001 P
    &EXIT
```

FORMAT 18 PROCESSING: RAMIS RUN PROCEDURE = M18RPA

FILENAME: M18RPA

FILETYPE: RUNPROC

```
DELETE
FILE RAMEXEC
1=OUT=M18RPA,$
END
CATALOG M18RPA
&& * CHANGE INSTRUCTOR ID PROCESSING;  INSTR FILE
&& * ENTRY ARGS ARE : (1) TRANS NUMBER ; (2) OLD ID, FIRST
&& * 6 CHARACTERS ; (3) OLD ID, LAST 3 CHARS ; (4) NEW ID,
&& * FIRST 6 CHARS ; (5) NEW ID, LAST 6 CHARS //
&& ARGS CONCAT=.
&& &ER = 'NO ERRORS' ;
&& PRINT SPACE
&& PRINT NOW PROCESSING TRANSACTION NUMBER &1
&& PRINT INSTR ID &2.&3 TO BE CHANGED TO &4.&5
DUMP
FILE INST
PRINT SLIB AND SAVE
IF ID IS &2.&3
END
&& RUN
&& IF &COND NE 0 THEN GO TO BUST ;
&& IF &RECORDS NE 0 THEN GO TO S1 ;
&&BUST
&& &ER = 'ERRORS WERE' ;
&&S1
&& PRINT SPACE
&& PRINT TRANSACTION NUMBER &1 - INSTRUCTOR FILE PROCESSING
&& PRINT &ER DETECTED IN STEP1 ; RAMIS INDICATORS ARE :
&& RPINT &COND , &NOKEYS , &REJECTS , &RECORDS ,
&& PRINT &INPUTS , &TRANS , &LINES
END CATALOG
```

FORMAT 18 PROCESSING: RAMIS RUN PROCEDURE = M18RPB

FILENAME: M18RPB

FILETYPE: RUNPROC

```
DELETE
FILE RAMEXEC
1=OUT=M18RPB,%
END
CATALOG M18RPB
&& * CHANGE AN INSTRUCTOR ID
&& * INSTRUCTOR FILE, 2ND STAGE PROCESSING
&& * ENTRY ARGS ARE : (1) TRANS NUMBER ; (2) OLD ID,
&& * FIRST 6 CHARS ; (3) OLD ID, LAST 3 CHARS ;
&& * (4) NEW ID, FIRST 6 CHARS ; (5) NEW ID, LAST 3 CHARS //
&& ARGS CONCAT=.
&& &ER = 'NO ERRORS' ;
&& &STEP = 'STEP3' ;
DELETE
FILE INST
ID=OUT=&2.&3,%
END
&& RUN
&& IF &COND NE 0 THEN GO TO BUST ;
INPUT
READ INST MDUMP2
RECORD 160
SET 1
FORM 9,16,2,5,4,4,4,4,4
FORM 4,4,4,4,4,4,4,4,4
FORM 4,4,4,4,4,4,4,4
ORDER 1,2,3,4,5,6,7,8,9,10,11,12
ORDER 13,14,15,16,17,18,19,20
ORDER 21,22,23,24,25,26
INSERT ID = &4.&5
FILE INST
END
&& RUN
&& &STEP = 'STEP4' ;
&& IF &COND NE 0 THEN GO TO BUST ;
&& IF &INPUTS NE 0 THEN GO TO S1 ;
&&BUST
&& &EP = 'ERRORS WERE' ;
&&S1
&& PRINT SPACE
&& PRINT TRANSACTION NUMBER &1 - INSTRUCTOR FILE PROCESSING
&& PRINT &EP DETECTED IN &STEP ; RAMIS INDICATORS ARE :
&& PRINT &COND , &NOKEYS , &REJECTS , &RECORDS ,
&& PRINT &CHANGES , &INPUTS , &TRANS , &LINES
END CATALOG
```

FORMAT 18 PROCESSING: RAMIS RUN PROCEDURE = M18RPC

FILENAME: M18RPC

FILETYPE: RUNPROC

```
DELETE
FILE RAMEXEC
1=OUT=M18RPC,$
END
CATALOG M18RPC
&& * CHANGE AN INSTRUCTOR ID
&& * COURSE FILE PROCESSING
&& * ENTRY ARGS ARE : (1) TRANSACTION NUMBER ; (2) OLD
&& * ID, FIRST 6 CHARS ; (3) OLD ID, LAST 3 CHARS ;
&& * (4) NEW ID, FIRST 6 CHARS ; (5) NEW ID, LAST 3 CHARS //
&& ARGS CONCAT=,
&& &ER = 'NO ERRORS' ;
&& &STEP = 'STEP5' ;
TABLE
FILE COURSE
PRINT CDP AND F1 AND F2 AND F3 AND SAVE
IF F1 IS &2,&3
END
&& RUN
&& IF &COND NE 0 THEN GOTO BUST ;
&& IF &RECORDS EQ 0 THEN GO TO BUST ;
&& &STEP = 'STEP6' ;
DELETE
READ FILE FT04F001
RECORD 18
FORM 4,9,4,1
ORDER 1,53,54,55
SET 53
FILE COURSE
END
&& RUN
&& IF &COND NE 0 THEN GOTO BUST ;
&& IF &CHANGES EQ 0 THEN GOTO BUST ;
&& &STEP = 'STEP7' ;
REVISE
READ FILE FT04F001
RECORD 18
FORM 4,X9,4,1
ORDER 1,54,55
INSERT F1 = &4,&5
KEY 1,53
FILE COURSE
END
&& RUN
&& IF &COND NE 0 THEN GOTO BUST ;
&& IF &INPUTS NE 0 THEN GOTO S1 ;
&&BUST
&& &ER = 'ERRORS WERE' ;
&&S1
&& PRINT SPACE
&& PRINT TRANSACTION NUMBER &1 - COURSE FILE PROCESSING ONLY
&& PRINT &ER DETECTED IN &STEP ; RAMIS INDICATORS ARE :
&& PRINT &COND , &NOKEYS , &REJECTS , &RECORDS ,
&& PRINT &CHANGES , &INPUTS , &TRANS , &LINES
```

TRANSACTION FORMAT : 19

FUNCTION : Change an Existing Building/Room Number

DATA BASE(S) MODIFIED : Facilities File

TRANSACTION TYPE : General Type E
Multiple Step RAMIS Execution

TRANSACTION FILE : Filename/Filetype = M19EXA/EXEC

EXECUTION SEQUENCE : 10

ADDITIONAL SUPPORT : (1) M19EXB/EXEC - NCSS Executive
(2) M15EXC/EXEC - NCSS Executive
(3) M99EXA/EXEC - NCSS Executive
(4) M19RP/RUNPROC - RAMIS Run Procedure

A training facility's building and room number appear only in the DOTS Facilities file. Within this file, they are key fields for the second-level segment. The second-level segment is thus maintained in sorted order by these two fields.

Change of a key field must be accomplished by deleting the old segment and re-entry of the deleted segment with the new key value.

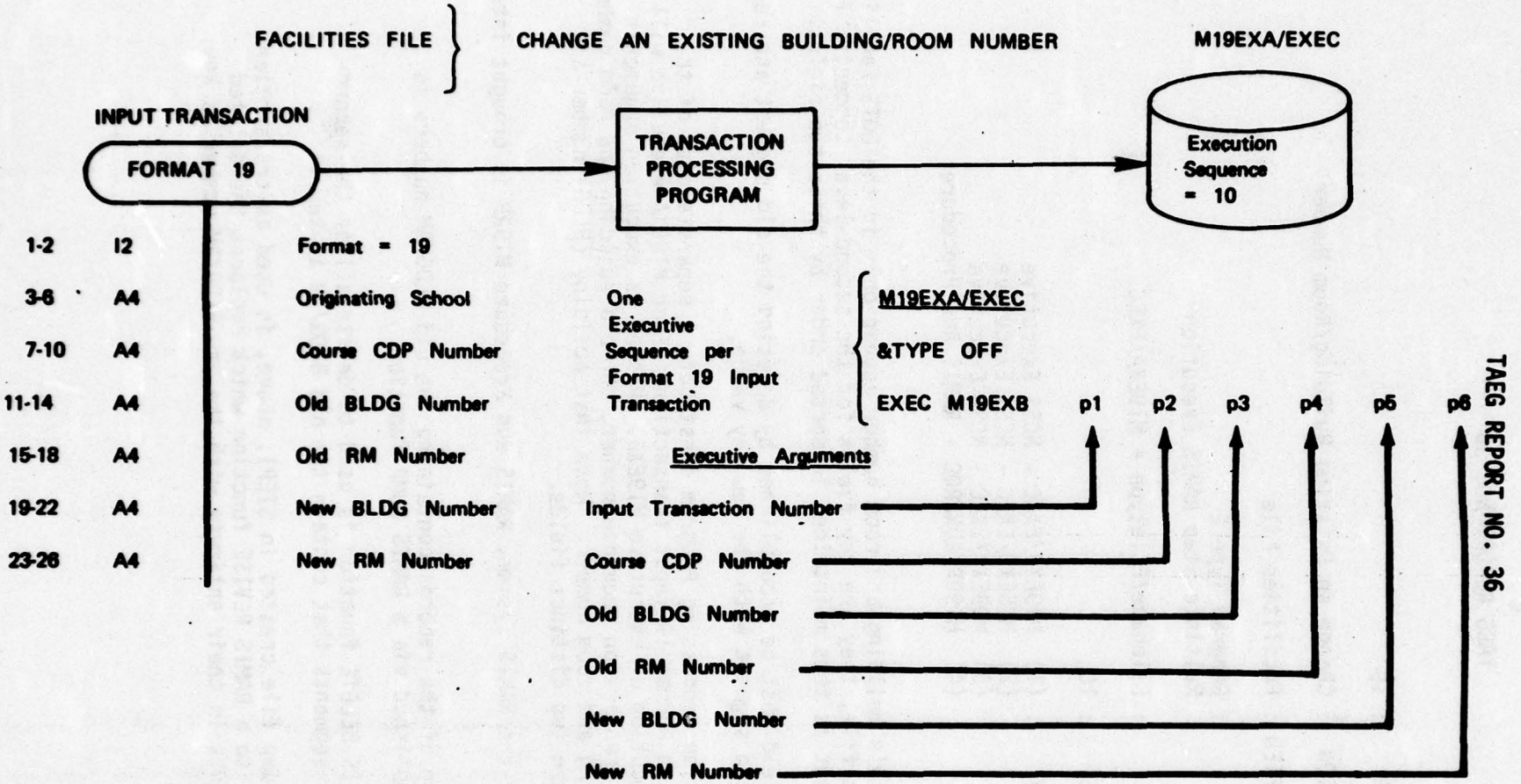
The Input Transaction Processing Program constructs a separate copy of the M19EXA/EXEC for each Format 19 input transaction. Each M19EXA executive will cause a unique execution of executive M19EXB. Passed as execution arguments will be the input transaction sequence number, the old building and room number, and the new building and room number. Note that building (BLDG) number and room (RM) number are two distinct fields.

After initializing the RAMIS system, RAMIS Run Procedure M19RP is brought into execution:

STEP1 - A dump of the records containing the old BLDG/RM numbers is accomplished via a RAMIS DUMP function.

STEP2 - A RAMIS DELETE function is used to delete all of the second-level segments that contain the old BLDG/RM as key.

STEP3 - The dump file created in STEP1, above, is used as transaction input to a RAMIS REVISE function which replaces the deleted segments in their entirety with the new BLDG/RM number as key.



FORMAT 19 INPUT TRANSACTION PROCESSING

FORMAT 19 PROCESSING: NCSS EXECUTIVE = M19EXB

FILENAME: M19EXB

FILETYPE: EXEC

```
    &TYPE OFF
    &COMMENT CHANGE A BLDG/RM. 2ND-LEVEL EXEC.
    &COMMENT ENTRY ARGS ARE :
    &COMMENT (1) TRANS NUMBER / (2) COP NUMBER /
    &COMMENT (3) OLD BLDG NUMBER / (4) OLD RM NUMBER /
    &COMMENT (5) NEW BLDG NUMBER / (6) NEW RM NUMBER //
STATE FACIL MDUMP P
    &IF &INDEX0 NE 0 &GOTO -S1
ERASE FACIL MDUMP P
-S1 FILEDEF RAMUTIL1 DSK FACIL MDUMP RE F LR 80 BL 80
    &STACK KT
    &STACK EXEC M19RP &1 . &2 . &3 .
    &STACK &4 . &5 . &6
    &STACK FIN
ATTACH RAMIS ALL BR
EXEC M15EXC
DETACH RAMIS BR
    &STACK RT
EXEC M99FXA &1
    &EXIT
```

FORMAT 19 PROCESSING: NCSS EXECUTIVE = M15EXC

FILENAME: M15EXC

FILETYPE: EXEC

```
    &TYPE OFF
    &COMMENT INTERNAL EXEC FOR DELETE SCHOOL.
    &COMMENT THIS EXEC MERELY PRINTS THE MSG/KILL FILE.
    &COMMENT NO ENTRY ARGS.
FILEDEF 6 DSK MSG KILL
EXEC RAMIS
    &STACK RT
FILEDEF 6 CONO
    &STACK KT
    &STACK L. /SPACE/
    &STACK C /SPACE//
    &STACK DOWN.
    &STACK P 4
    &STACK 6
EDIT MSG KILL
    &EXIT
```

FORMAT 19 PROCESSING: NCSS EXECUTIVE = M99EXA

FILENAME: M99EXA

FILETYPE: EXEC

```
    &TYPE OFF
    &COMMENT G.P. END TRANS PROCESSING MSG
    &COMMENT SINGLE ARG IS : (1) TRANSACTION NUMBER
    &SPACE
    &PRINT TRANSACTION &1 PROCESSING COMPLETED
STATE FILE FT04F001 P
    &IF &INDEX0 NE 0 &EXIT
ERASE FILE FT04F001 P
    &EXIT
```

FORMAT 19 PROCESSING: RAMIS RUN PROCEDURE = M19RP

FILENAME: M19RP

FILETYPE: RUNPROC

```
DELETE
FILE RAMEXEC
1=OUT=M19RP.$
END
CATALOG M19RP
&& &ER = 'NO ERRORS' ;
&& &STEP = 'STEP1' ;
DUMP
FILE FACIL
PRINT AVHRS AND SAVE
IF CDP IS &2
IF BLDG IS &3
IF RM IS &4
END
&& RUN
&& IF &COND NE 0 THEN GOTO BUST ;
&& IF &RECORDS EQ 0 THEN GOTO BUST ;
&& &STEP = 'STEP2' ;
DELETE
FILE FACIL
CDP=&2,BLDG=&3,RM=OUT=&4.$
END
&& RUN
&& IF &COND NE 0 THEN GOTO BUST ;
&& IF &CHANGES EQ 0 THEN GOTO BUST ;
&& &STEP = 'STEP3' ;
REVISE
READ FACIL MDUMP
RECORD 80
FORM 4,X12.3,1,6,5
ORDER 1,5,6,7,8
KEY 1,3,4
INCLUDE 3
INSERT BLDG = &5
INSERT RM = &6
FILE FACIL
END
&& IF &COND NE 0 THEN GOTO BUST ;
&& IF &INPUTS NE 0 THEN GOTO S1 ;
&&BUST
&& &ER = 'ERRORS WERE' ;
&&S1
&& PRINT SPACE
&& PRINT TRANSACTION NUMBER &1 - FACILITIES FILE ONLY
&& PRINT &ER DETECTED IN &STEP ; RAMIS INDICATORS ARE :
&& PRINT &COND , &NOKEYS , &REJECTS , &RECORDS ,
&& PRINT &CHANGES , &INPUTS , &TRANS , &LINES
END CATALOG
```

TRANSACTION FORMAT : 20

FUNCTION : Change a School Code

DATA BASE(S) MODIFIED: Facilities File
Instructor File
Course File

TRANSACTION TYPE : General Type E
Multiple Step RAMIS Execution

TRANSACTION FILE : Filename/Filetype = M20EXA/EXEC

EXECUTION SEQUENCE : 11

ADDITIONAL SUPPORT :
(1) M20EXB/EXEC - NCSS Executive
(2) M15EXC/EXEC - NCSS Executive
(3) M99EXA/EXEC - NCSS Executive
(4) M20RPA/RUNPROC - RAMIS Run Procedure
(5) M20RPB/RUNPROC - RAMIS Run Procedure
(6) M20RPC/RUNPROC - RAMIS Run Procedure

Although school is not a key field, all of the records of the three DOTS data bases contain a school code. Thus, a Format 20 input transaction will effect multiple changes in all three of the DOTS files.

Since school is not key, a single RAMIS CHANGE function directed at each of the three DOTS files in turn is sufficient to cause the desired modifications. All records bearing the old school code will be changed to reflect the new code.

The Input Transaction Processing Program constructs a separate copy of the M20EXA executive on a one-for-one basis with each Format 20 input transaction. Each M20EXA executive will then, in turn, cause one separate execution of the M20EXB second-level executive.

After RAMIS system initialization, executive M20EXB passes control to the following three RAMIS Run Procedures (in the sequence shown):

Run Procedure M20RPA

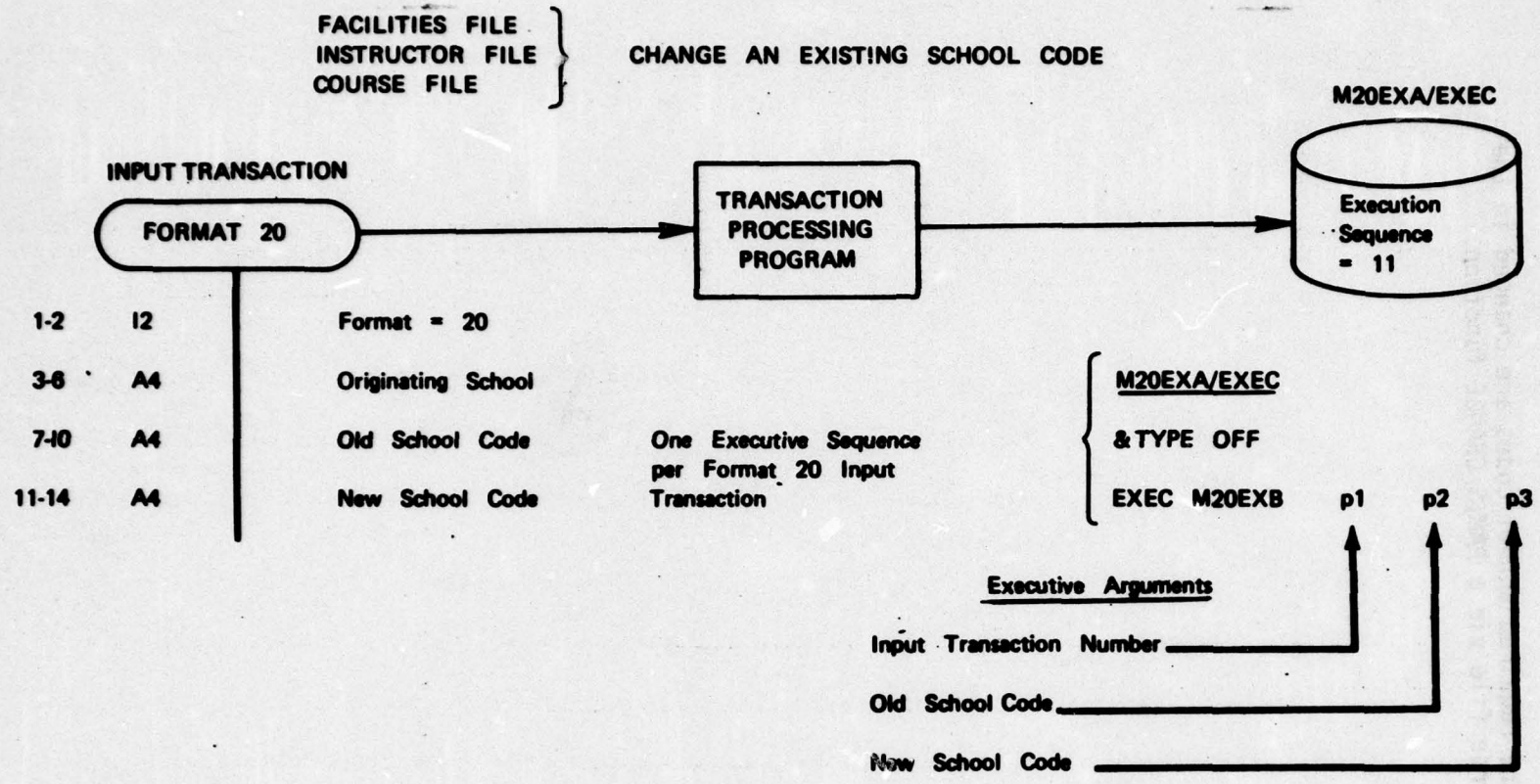
STEP1 - A RAMIS CHANGE function is directed at the DOTS Facilities file to effect the global change of the school code fields. All records tagged with the old school code are modified to reflect the new school code.

Run Procedure M20RPB

STEP2 - A RAMIS CHANGE function is used to change all of the required school codes in the DOTS Instructor file.

Run Procedure M20RPC

STEP3 - All of the required school codes are changed in the DOTS Course file via a RAMIS CHANGE function.



FORMAT 20 INPUT TRANSACTION PROCESSING

FORMAT 20 PROCESSING: NCSS EXECUTIVE = M20EXB

FILENAME: M20EXB

FILETYPE: EXEC

```
     &TYPE OFF
     &COMMENT CHANGE A SCHOOL NAME. 2ND LEVEL EXEC.
     &COMMENT ENTRY ARGS ARE:
     &COMMENT (1) TRANSACTION NUMBER /
     &COMMENT (2) OLD SCHOOL CODE / (3) NEW SCHOOL CODE //
     &STACK RT
     &STACK EXEC M20PPA &1 . &2 . &3
     &STACK FIN
ATTACH RAMIS ALL BR
EXEC M15EXC
     &STACK EXEC M20PPB &1 . &2 . &3
     &STACK FIN
EXEC M15EXC
     &STACK EXEC M20PPC &1 . &2 . &3
     &STACK FIN
EXEC M15EXC
DETACH RAMIS BR
     &STACK RT
EXEC M99EXA &1
     &EXIT
```

FORMAT 20 PROCESSING: NCSS EXECUTIVE = M15EXC

FILENAME: M15EXC

FILETYPE: EXEC

```
    &TYPE OFF
    &COMMENT INTERNAL EXEC FOR DELETE SCHOOL.
    &COMMENT THIS EXEC MERELY PRINTS THE MSG/KILL FILE.
    &COMMENT NO ENTRY ARGS.
FILEDEF 6 DSK MSG KILL
EXEC RAMIS
    &STACK RT
FILEDEF 6 CONO
    &STACK KT
    &STACK L. /SPACE/
    &STACK C /SPACE//
    &STACK DOWN.
    &STACK P 4
    &STACK G
EDIT MSG KILL
    &EXIT
```

FORMAT 20 PROCESSING: NCSS EXECUTIVE = M99EXA

FILENAME: M99EXA

FILETYPE: EXEC

```
    &TYPE OFF
    &COMMENT G.P. END TRANS PROCESSING MSG
    &COMMENT SINGLE ARG IS : (1) TRANSACTION NUMBER
    &SPACE
    &PRINT TRANSACTION &1 PROCESSING COMPLETED
STATE FILE FT04F001 P
    &IF &INDEX0 NE 0 &EXIT
ERASE FILE FT04F001 P
    &EXIT
```

FORMAT 20 PROCESSING: RAMIS RUN PROCEDURE = M20RPA

FILENAME: M20RPA

FILETYPE: RUNPROC

```
DELETE
FILE RAMEXEC
1=OUT=M20RPA,$
END
CATALOG M20RPA
&& &ER = 'NO ERRORS' :
&& &STEP = 'STEP1' :
CHANGE
FILE FACIL
SCH=&2,$
SCH=SET=&3,$
END
&& RUN
&& IF &COND NE 0 THEN GOTO &UST :
&& IF &CHANGES NE 0 THEN GOTO S1 :
&&&UST
&& &ER = 'ERRORS WERE' :
&&S1
&& PRINT SPACE
&& PRINT TRANSACTION NUMBER &1 - FACILITIES FILE PROCESSING
&& PRINT &ER DETECTED IN &STEP : RAMIS INDICATORS ARE :
&& PRINT &COND , &NOKEYS , &REJECTS , &RECORDS ,
&& PRINT &CHANGES , &INPUTS , &TRANS , &LINES
END CATALOG
```

FORMAT 20 PROCESSING: RAMIS RUN PROCEDURE = M20RPB

FILENAME: M20RPB

FILETYPE: RUNPROC

```
DELETE
FILE RAMEXEC
1=OUT=M20RPB,$
END
CATALOG M20RPB
&& &EP = 'NO ERRORS' ;
&& &STEP = 'STEP2' ;
CHANGE
FILE INST
SCH=&2,$
SCH=SET=&3,$
END
&& RUN
&& IF &COND NE 0 THEN GOTO BUST ;
&& IF &CHANGES NE 0 THEN GOTO S1 ;
&&BUST
&& &ER = 'ERRORS WERE' ;
&&S1
&& PRINT SPACE
&& PRINT TRANSACTION NUMBER &1 - INSTRUCTOR FILE PROCESSING
&& PRINT &ER DETECTED IN &STEP ; RAMIS INDICATORS ARE :
&& PRINT &COND , &NOKEYS , &REJECTS , &RECORDS ,
&& PRINT &CHANGES , &INPUTS , &TRANS , &LINES
END CATALOG
```

FORMAT 20 PROCESSING: RAMIS RUN PROCEDURE = M20RPC

FILENAME: M20RPC

FILETYPE: RUNPROC

```
DELETE
FILE RAMEXEC
1=OUT=M20RPC,$
END
CATALOG M20RPC
&& &ER = 'NO ERRORS' ;
&& &STEP = 'STEP2' ;
CHANGE
FILE COURSE
SCH=&2,$
SCH=SET=&3,$
END
&& RUN
&& IF &COND NE 0 THEN GOTO &UST ;
&& IF &CHANGES NE 0 THEN GOTO S1 ;
&&&UST
&& &ER = 'ERRORS WERE' ;
&&S1
&& PRINT SPACE
&& PRINT TRANSACTION NUMBER &1 - COURSE FILE PROCESSING
&& PRINT &ER DETECTED IN &STEP ; RAMIS INDICATORS ARE :
&& PRINT &COND , &NOKEYS , &REJECTS , &RECORDS ,
&& PRINT &CHANGES , &INPUTS , &TRANS , &LINES
END CATALOG
```

TRANSACTION FORMAT : 21

FUNCTION : Add a New Course

DATA BASE(S) MODIFIED : Course File

TRANSACTION TYPE : General Type D
Single Step RAMIS Execution

TRANSACTION FILE : Filename/Filetype = DSB/RAM

EXECUTION SEQUENCE : 12

This transaction type serves to establish a new course to the DOTS Course file. The associated school code, course CDP number, the CIN number, and course title are provided with the input transaction. Once the course record has been established, the other fields of the Course file can be added via the following input transactions directed explicitly at the Course file:

Format 6: Change a non-key level-1 field

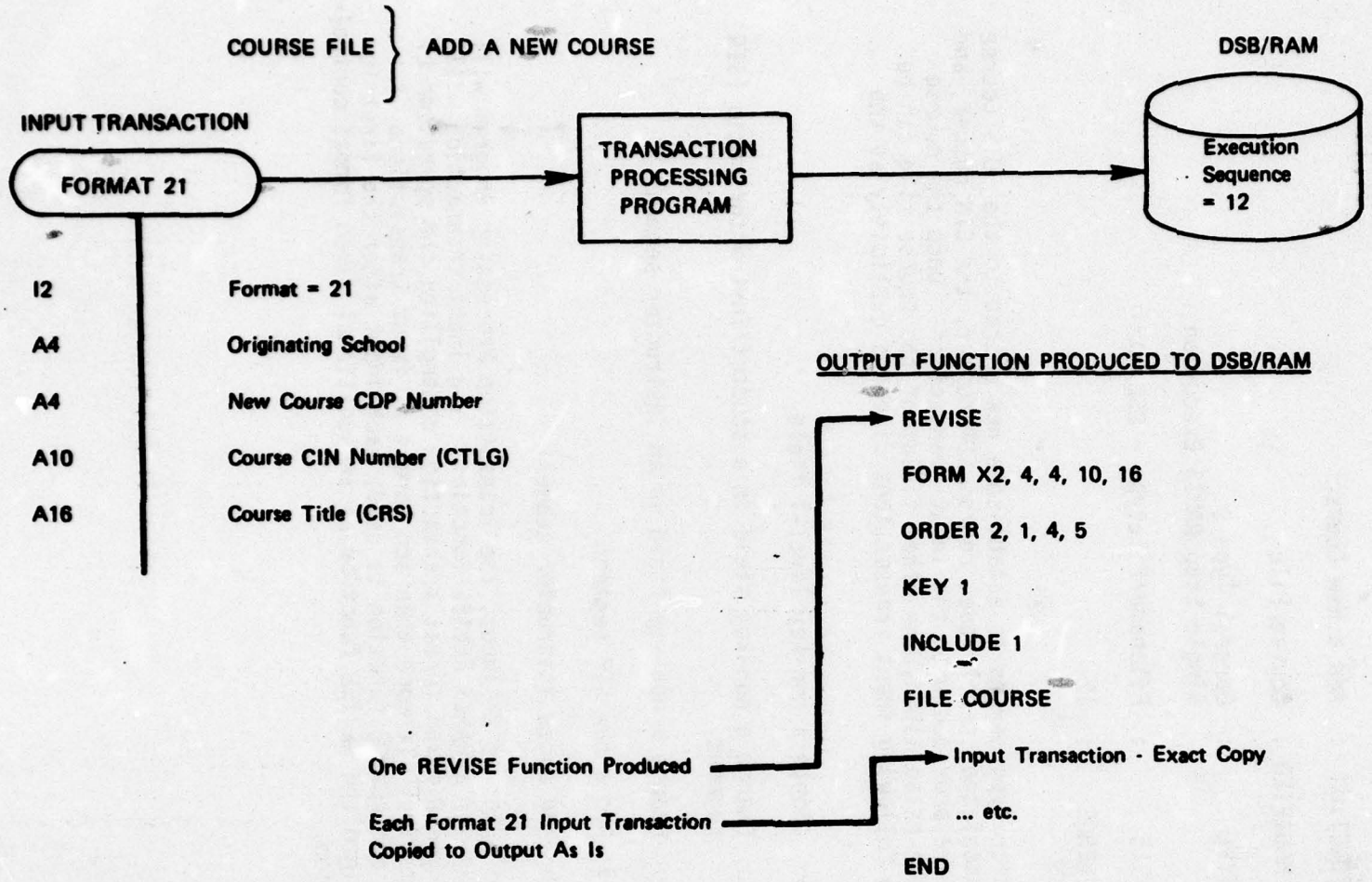
Format 7: Change a non-key field in a student/instructor ratio (SIR) segment

Format 8: Change a non-key field in an instructor segment

Format 23: Add a new SIR segment

Format 26: Add a new instructor segment

In response to a Format 21 input, the Transaction Processing Program will produce one copy of a RAMIS REVISE function. The input transaction will be used as a type D fixed format transaction controlling the operation of the REVISE function. If more than one Format 21 input transactions is present, only one REVISE function is used; each input after the first is incorporated into the REVISE function as an additional fixed format controlling transaction.



FORMAT 21 INPUT TRANSACTION PROCESSING

TRANSACTION FORMAT : 23

FUNCTION : Add a New Student/Instructor Ratio (SIR) Segment to the Course File

DATA BASE(S) MODIFIED : Course File

TRANSACTION TYPE : General Type D
Single Step RAMIS Execution

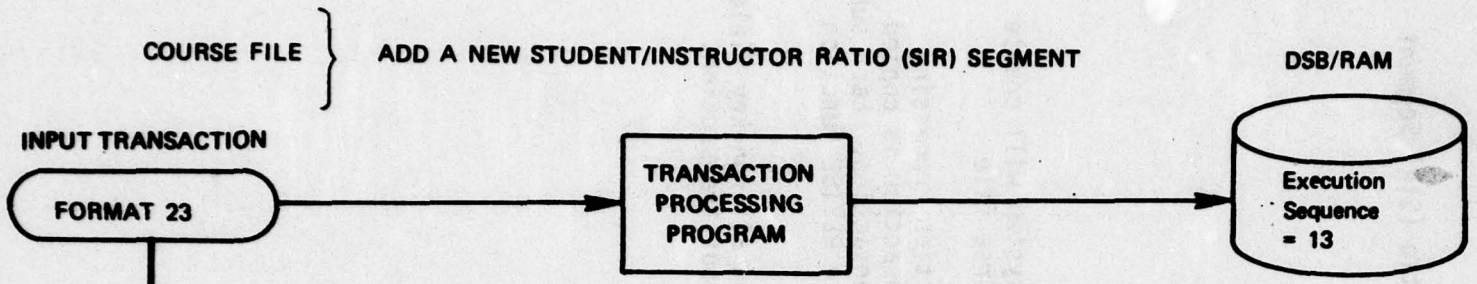
TRANSACTION FILE : Filename/Filetype = DSB/RAM

EXECUTION SEQUENCE : 13

In response to a Format 23 transaction, the maintenance system will create and append a new SIR segment to a given course in the Course file.

A single RAMIS REVISE function is produced by the Transaction Processing Program as a result of a Format 23 input. The input transaction is copied (exactly) into the REVISE function as the controlling transaction. Each subsequent Format 23 input (if any) is copied into the single REVISE function in a similar fashion.

For the SIR second level segment, the key field (SIR), and all non-key fields (contract hours and course type) are provided with the input transaction. The controlling course CDP number is also provided.



1-2	I2	Format = 23
3-6	A4	Originating School
7-10	A4	Course CDP Number
11-13	I3	New Student/Instructor Ratio (SIR)
14-17	I4	Course Contact Hours (CHRS)
18-18	I1	Course Type (CTYP)

OUTPUT FUNCTION TO BE PLACED IN DSB/RAM

One REVISE Function Produced

Each Format 23 Input Transaction Copied to Output

```
REVISE
FORM X6, 4, 3, 4, 1
ORDER 1, 53, 54, 55
KEY 1, 53
INCLUDE 53
FILE COURSE
    Input Transaction - Exact Copy
    ... etc.
END
```

FORMAT 23 INPUT TRANSACTION PROCESSING

TRANSACTION FORMAT : 24

FUNCTION : Add a New Instructor to the Instructor File

DATA BASE(S) MODIFIED : Instructor File

TRANSACTION TYPE : General Type D
Single Step RAMIS Execution

TRANSACTION FILE : Filename/File Type = DSB/RAM

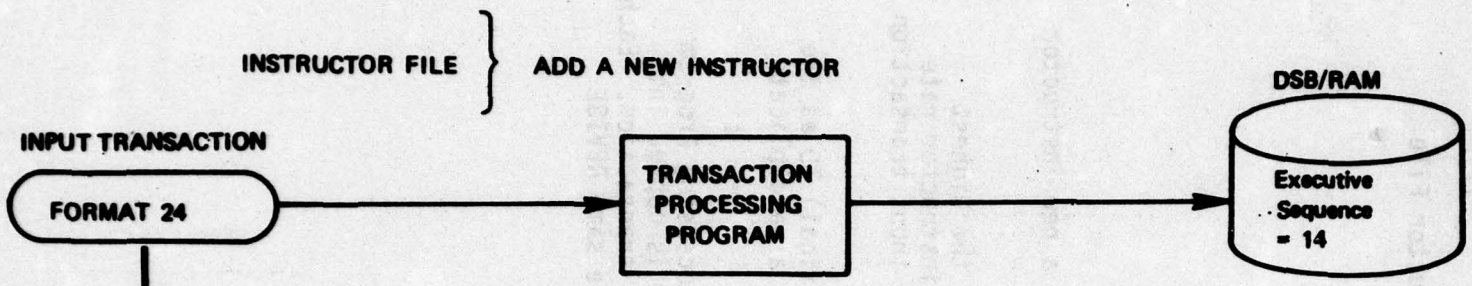
EXECUTION SEQUENCE : 14

A Format 24 input transaction commands the introduction of a new instructor record into the DOTS Instructor file.

Instructor ID (Social Security Number) is the key field on the highest level in the Instructor file. This key value, as well as instructor rate (RATE), name (NME), and initials (IN) are provided on the input transaction and are entered into the file with the new record.

If an existing Instructor file record bears the same (identical) ID as the incoming transaction, the new transaction will be rejected as a duplicate record error.

A single REVISE function is produced by the Transaction Processing Program as a response to a Format 24 input. The input transaction is copied into the REVISE function as a standard fixed format controlling transaction. Each subsequent Format 24 input (if any) will be copied into the same REVISE function.



1-2	I2	Format = 24
3-6	A4	Originating School
7-15	I9	New Instructor ID
16-20	A5	Instructor RATE
21-22	A2	Instructor Initials (IN)
23-38	A16	Instructor Surname (NME)

OUTPUT FUNCTION PRODUCED TO DSB/RAM

REVISE

FORM X2, 4, 9, 5, 2, 16

ORDER 5, 1, 4, 3, 2

KEY 1

INCLUDE 1

FILE INST

Only One REVISE Function Produced

Each Format 24 Input Transaction Copied to Output without Modification → Input Transaction - Exact Copy

... etc.

END

FORMAT 24 INPUT TRANSACTION PROCESSING

TRANSACTION FORMAT : 25

FUNCTION : Add a New Building/Room Segment to the Facilities File

DATA BASE(S) MODIFIED : Facilities File

TRANSACTION TYPE : General Type D
Single Step RAMIS Execution

TRANSACTION FILE : Filename/Filetype = DSB/RAM

EXECUTION SEQUENCE : 15

As a result of a Format 25 input transaction, a new building/room (BLDG/RM) lower level segment will be added to the DOTS Facilities file. All of the key and non-key fields required to establish a new BLDG/RM segment are provided with the input transaction.

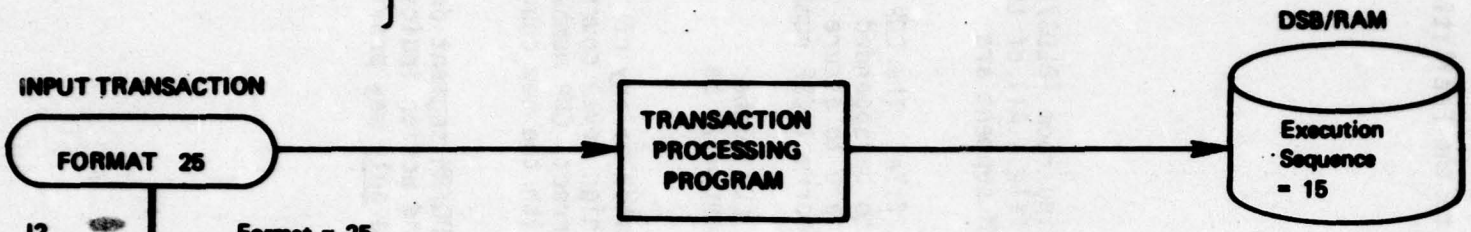
Course CDP number is the high-order key in the Facilities file. The CDP number of the course to which the new BLDG/RM segment is to be appended is provided on the input transaction. The user should take care to assure that the CDP number provided is correct. The results of an incorrect CDP number are:

- If the incorrect number matches an existing legal CDP number in the Facilities file, then the new BLDG/RM segment will be appended to that course.

- If the incorrect CDP number is unique, i.e., no match on any of the existing CDP numbers in the file, then a new high level course segment will be created tagged with the new (incorrect) CDP number. The new BLDG/RM segment will then be associated with the new course record.

If it is known that the course CDP associated with the BLDG/RM segment does not presently exist in the Facilities file, then the second action indicated above would be considered normal. At present, this is the only way provided to add a new CDP to the Facilities file.

FACILITIES FILE } ADD A NEW BUILDING/ROOM SEGMENT



1-2	I2	Format = 25
3-6	A4	Originating School
7-10	A4	Course CDP Number
11-14	A4	New Building Number (BLDG)
15-18	A4	New Room Number (RM)
19-21	I3	Room Capacity (RMCAP)
22-22	A1	Room Type (RMTP)
23-29	F6.1	Required Hours (REQHRS) (Maximum Allowed is 9,999.9 Hours)
30-34	I5	Available Hours (AVHRS)

OUTPUT FUNCTION WRITTEN TO DSB/RAM

One REVISE Function Produced

REVISE

FORM X2, 4, 4, 4, 4, 3, 1, 7, 5

ORDER 2, 1, 3, 4, 5, 6, 7, 8

KEY 1, 3, 4

INCLUDE 1

FILE FACIL

Each Format 25 Input Transaction Copied to Output Without Modification

Input Transaction - Exact Copy

... etc.

END

FORMAT 25 INPUT TRANSACTION PROCESSING

TRANSACTION FORMAT : 26

FUNCTION : Add an Instructor Segment to the Course File

DATA BASE(S) MODIFIED : Course File

TRANSACTION TYPE : General Type D
Single Step RAMIS Execution

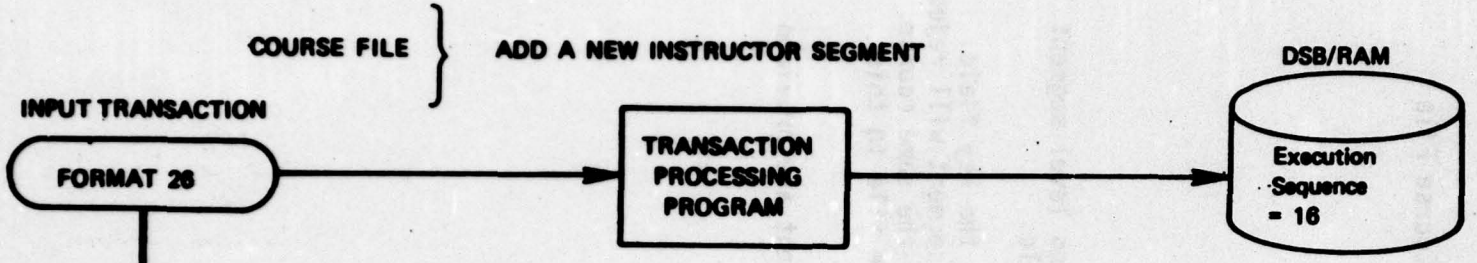
TRANSACTION FILE : Filename/Filetype = DSB/RAM

EXECUTION SEQUENCE : 16

A Format 26 input transaction causes a new instructor second level segment to be added to a given course record in the DOTS Course file.

For the instructor segment, ID (Social Security Number) is the key field. The maintenance operation of appending the new instructor segment will reject an ID that is a duplicate of an existing instructor ID for the same course. The new (duplicate) segment will not be introduced into the file in this case.

All of the fields, key and non-key, of the instructor segment are provided with the input transaction.



1-2	I2	Format = 26
3-6	A4	Originating School
7-10	A4	Course CDP Number
11-19	I9	New Instructor ID
20-23	I4	Instructor Qualification (QUAL)
24-24	I1	Instructor Assignment Flag (ASN)

OUTPUT FUNCTION WRITTEN TO DSB/RAM

Only One REVISE Function Produced

```
REVISE
FORM X6, 4, 9, 4, 1
ORDER 1, 53, 54, 55
KEY 1, 53
INCLUDE 53
FILE COURSE
```

Each Format 26 Input Transaction Copied to Output Without Modification → Input Transaction - Exact Copy

```
... etc.
END
```

FORMAT 26 INPUT TRANSACTION PROCESSING

TRANSACTION FORMAT : 27

FUNCTION : Delete an Instructor Segment from the Course File

DATA BASE(S) MODIFIED : Course File

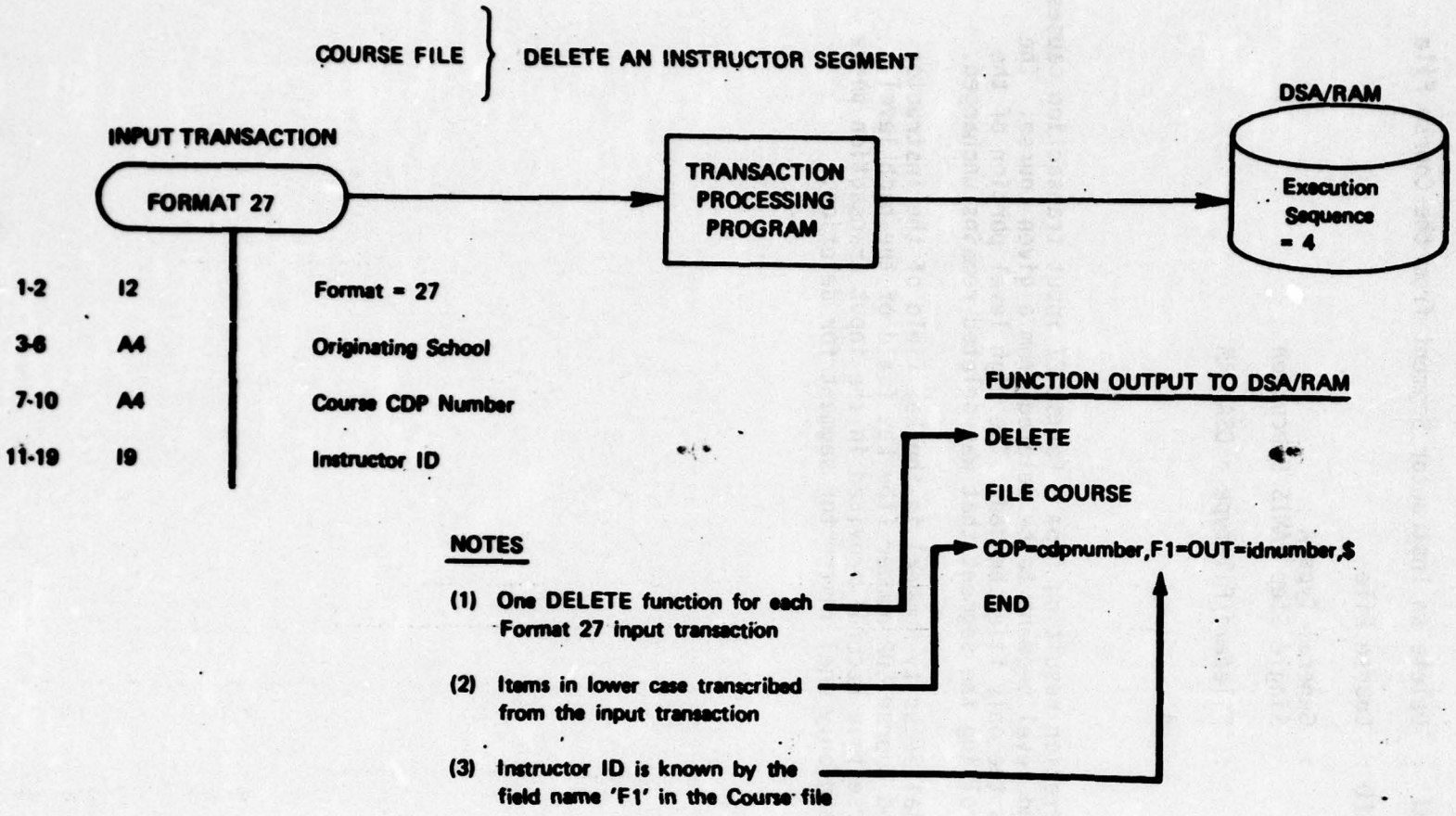
TRANSACTION TYPE : General Type A
Single Step RAMIS Execution

TRANSACTION FILE : Filename/Filetype = DSA/RAM

EXECUTION SEQUENCE : 4

The maintenance operation resulting from a Format 27 input transaction causes an instructor second level segment to be deleted from a given course. The DOTS Course file is the only file changed. The high level portion of the course record controlling the segment that was deleted remains unchanged.

Instructor ID (Social Security Number) is the key field of the instructor segment. The ID and course CDP number (the key field of the high level portion of the Course file record) provided in the input transaction serve to delineate one (and only one) instructor segment for deletion.



FORMAT 27 INPUT TRANSACTION PROCESSING

NCSS PLI PROGRAMS

DOTS MAINTENANCE PROGRAM: Filename = FCOURSE
Filetype = PLI

PROGRAMMING LANGUAGE : PL/1

NCSS EXECUTIVES : FCOURSE

PROGRAM FUNCTION

This program produces a printer output to assist the user in the maintenance of the DOTS Course file. The blank forms produced have been designed with spacing sufficient to allow entry of handwritten data.

PROGRAM FCOURSE OUTPUT

A sample output from program FCOURSE follows. Two unique courses may be mentioned per page. The program produces ten (10) such pages for each run.

At the time of this document, the executive FCOURSE forces routing of the forms output of this program to the NCSS San Diego office.

Listing of Filename = FCOURSE, Filetype = PLI

```
FCOURSE:      /* MAKEUP COURSE BLANK FORMS - MON/19/JUL/76 */
        PROC OPTIONS (MAIN) ;

DECLARE T1    CHAR (58) STATIC INIT (
'DOTS COURSE FILE (FILENAME = COURSE) BLANK SUBMISSION FORM' ),
        T5    CHAR (132) STATIC INIT (
'             -------          -------          -------              ----
--            -------          -------    ---------  -----  ---' ),
        NPAGE      FIXED BIN STATIC INIT ( 10 ),
        T3    CHAR (132) STATIC INIT (
'CDP   SCH       CTLG              CRS          CTYPE  PNEC  SUIC   RDTE
  CCMM      PRTY                                SIR/      CHRS/  TY/' ),
        T4    CHAR (132) STATIC INIT (
'----  ----  ----------  ----------------  -----  ----  -----  ------  -
----------  ----                          ---------  -----  ---' ) ;
START:  NC = 0 ;
        OPEN FILE (SYSPRINT)
             LINESIZE (132) PAGESIZE (58) ;

        ON ENDPAGE (SYSPRINT)
             BEGIN ;
             NC = NC + 1 ;
             IF ( NC > NPAGE ) THEN GO TO EOJ ;

             PUT EDIT ( T1, 'PAGE :', 'DATE :', 'ORIGINATOR :' )
                  ( PAGE, LINE(3), A(58),
                    SKIP(2), COL(19), A, COL(50), A, COL(80), A ) ;
```

Listing of Filename = FCOURSE, Filetype = PLI (Cont'd)

```
            PUT EDIT ( T3, 'ID       PQUAL  ASN', T4 )
                 ( SKIP(2), A(132), SKIP(1), COL(115), A,
                   SKIP(1), A(132) ) ;

            END ;

        SIGNAL ENDPAGE (SYSPRINT) ;

LOOP:   IF ( LINENO(SYSPRINT) > 34 )
            THEN SIGNAL ENDPAGE (SYSPRINT) ;

        PUT SKIP (5) ;

        PUT EDIT ( T4 ) ( SKIP(1), A(132) ) ;

        PUT EDIT ( 'CONV1 :', 'CONV2 :', 'CONV3 :', 'MAXCAP :',
             'PFAIL  :', 'HSETB  :', T5 ) ( R(R3) ) ;

R3:     FORMAT ( SKIP(2), COL(7), A, COL(24), A, COL(41), A,
             COL(58), A, COL(75), A, COL(93), A,
             SKIP(1), A(132) ) ;

        PUT EDIT ( 'LEN1  :', 'LEN2  :', 'LEN3  :', 'MINCAP :',
             'NADIS  :', 'TINSTR :', T5 ) ( R(R3) ) ;

        PUT EDIT ( 'CAP1  :', 'CAP2  :', 'CAP3  :', 'PCAP   :',
             'NOSHOW :', 'UDEM   :', T5 ) ( R(R3) ) ;

        PUT EDIT ( 'BCAP1 :', 'BCAP2 :', 'BCAP3 :', 'ECAP   :',
             'SETB   :', 'OPT1   :', T5 ) ( R(R3) ) ;

        PUT EDIT ( 'DMD1  :', 'DMD2  :', 'DMD3  :', 'SCAP   :',
             'BKLOG  :', 'OPT2   :', T5 ) ( R(R3) ) ;

        PUT EDIT ( 'BDMD1 :', 'BDMD2 :', 'BDMD3 :', 'UCAP   :',
             'PAOB   :', 'OPT3   :', T5 ) ( R(R3) ) ;

                GO TO LOOP ;

EOJ:    CLOSE FILE (SYSPRINT) ;

            END FCOURSE ;
```

Program FCOURSE Blank Form Output

UOTS COURSE FILE (FILENAME = COURSE) BLANK SUBMISSION FORM

PAGE : DATE : ORIGINATOR :

CDP	SCH	CTLG	CHS	CTYPE	PNEC	SUIC	RDTE	CCHM	PRTY				SIR/ ID	CHRS/ PQUAL	TY/ ASN
----	----	----------	----------	-----	----	-----	------	----------	----				---------	-----	--
----	----	----------	----------	-----	----	-----	------	----------	----				---------	-----	--
	CONV1 :	-------	CONV2 :	-------	CONV3 :	-------	MAXCAP :	------	PFAIL :	-------	HSETB :	-------	---------	-----	--
	LEN1 :	-------	LEN2 :	-------	LEN3 :	-------	MINCAP :	------	NADIS :	-------	TINSTR :	-------	---------	-----	--
	CAP1 :	-------	CAP2 :	-------	CAP3 :	-------	PCAP :	------	NOSHOW :	-------	UDEM :	-------	---------	-----	--
	BCAP1 :	-------	BCAP2 :	-------	BCAP3 :	-------	ECAP :	------	SETB :	-------	OPT1 :	-------	---------	-----	--
	DMD1 :	-------	DMD2 :	-------	DMD3 :	-------	SCAP :	------	BKLOG :	-------	OPT2 :	-------	---------	-----	--
	BDMD1 :	-------	BDMD2 :	-------	BDMD3 :	-------	UCAP :	------	PAOB :	-------	OPT3 :	-------	---------	-----	--
----	----	----------	----------	-----	----	-----	------	----------	----				---------	-----	--
	CONV1 :	-------	CONV2 :	-------	CONV3 :	-------	MAXCAP :	------	PFAIL :	-------	HSETB :	-------	---------	-----	--
	LEN1 :	-------	LEN2 :	-------	LEN3 :	-------	MINCAP :	------	NADIS :	-------	TINSTR :	-------	---------	-----	--
	CAP1 :	-------	CAP2 :	-------	CAP3 :	-------	PCAP :	------	NOSHOW :	-------	UDEM :	-------	---------	-----	--
	BCAP1 :	-------	BCAP2 :	-------	BCAP3 :	-------	ECAP :	------	SETB :	-------	OPT1 :	-------	---------	-----	--
	DMD1 :	-------	DMD2 :	-------	DMD3 :	-------	SCAP :	------	BKLOG :	-------	OPT2 :	-------	---------	-----	--
	BDMD1 :	-------	BDMD2 :	-------	BDMD3 :	-------	UCAP :	------	PAOB :	-------	OPT3 :	-------	---------	-----	--

DOTS MAINTENANCE PROGRAM: Filename = FINST
Filetype = PLI

PROGRAMMING LANGUAGE : PL/1

NCSS EXECUTIVES : FINST

PROGRAM FUNCTION

To assist the user in accumulating handwritten data changes for the DOTS Instructor file, a blank form has been designed. This program produces these forms as printer output.

PROGRAM FINST OUTPUT

Following is an illustration of the output from program FINST. One sheet allows reference to three distinct instruction records. Ten (10) pages of such forms are produced in a single run of FINST.

The executive FINST forces routing of the output of program FINST to the NCSS San Diego office.

Listing of Filename = FINST, Filetype = PLI

```
FINST:  /* PRODUCE INST FILE FILLIN FORMS - MON/19/JUL/76 */
        PROC OPTIONS (MAIN) ;

DECLARE T1    CHAR (132) STATIC,
        T2    CHAR (104) STATIC,
        T3    CHAR (84) STATIC,

        T4    CHAR (58) STATIC INIT (
'DOTS COURSE FILE (FILENAME = COURSE) BLANK SUBMISSION FORM' ),

        NPAGE     FIXED BIN STATIC INIT ( 10 ) ;

START:  NC = 0 ;
        T1 = '------------  -------  ---  --------------------'
             || (6)'  -------' ;

        T2 = '    ' || (5)'            --------' ;

        T3 = SUBSTR ( T2, 1, 84 ) ;

        OPEN FILE (SYSPRINT)
             LINESIZE (132) PAGESIZE (58) ;

        ON ENDPAGE (SYSPRINT)
             BEGIN ;
             NC = NC + 1 ;
             IF ( NC > NPAGE ) THEN GO TO EOJ ;
```

Listing of Filename = FINST, Filetype = PLI (Cont'd)

```
                PUT EDIT ( T4, 'PAGE :', 'DATE :', 'ORIGINATOR :' )
                     ( PAGE, LINE(3),
                       A(58), SKIP(2), COL(19), A, COL(50), A,
                       COL(80), A ) ;

                PUT EDIT ( 'ID', 'RATE    IN', 'NME', 'SCH',
                           'REP', 'ROT', 'PNEC', 'SNEC',
                           'NEC3', 'NEC4', 'NEC5', T1 )
                     ( SKIP(2),
                       COL(6), A, COL(18), A, COL(42), A, COL(58), A,
                       COL(68), A, COL(78), A,
                       X(1), 5 ( A(6), A(4) ),
                       SKIP(1), A(132) ) ;

                END ;

           SIGNAL ENDPAGE (SYSPRINT) ;

LOOP:      IF ( LINENO(SYSPRINT) > 44 )
                THEN SIGNAL ENDPAGE (SYSPRINT) ;

           PUT SKIP (5) ;

L1:        PUT EDIT ( T1 ) ( SKIP(1), A(132) ) ;

L2:        PUT EDIT ( 'CONT :', 'PREP :', 'FTRN :', 'FLET :', 'SKLV :',
                T2, 'CDEV :', 'BKIN :', 'MDUT :', 'ADMN :', 'SLIB :',

                T2, 'EQMT :', 'CTRN :', 'ISCH :', 'ANLV :',
                T3 )
                ( SKIP(2), COL(10), A, COL(30), A, COL(50), A, COL(70),
                     A, COL(90), A, SKIP(1), A(104),
                  SKIP(2), COL(10), A, COL(30), A, COL(50), A, COL(70),
                     A, COL(90), A, SKIP(1), A(104),
                  SKIP(2), COL(10), A, COL(30), A, COL(50), A, COL(70),
                     A, SKIP(1), A(84) ) ;

                GO TO LOOP ;

EOJ:       CLOSE FILE (SYSPRINT) ;

                END FINST ;
```

Program FINST Blank Form Output

DOTS INSTRUCTOR FILE (FILENAME = INST) BLANK SUBMISSION FORM

PAGE : DATE : ORIGINATOR :

ID	RATE	IN	NME	SCH	REP	ROT	PNEC	SNEC	NEC3	NEC4	NEC5

CONT : PREP : FTRN : FLET : SKLV :

CDEV : BKIN : MOUT : ADMN : SLIB :

EQMT : CTRN : ISCH : ANLV :

CONT : PREP : FTRN : FLET : SKLV :

CDEV : BKIN : MOUT : ADMN : SLIB :

EQMT : CTRN : ISCH : ANLV :

CONT : PREP : FTRN : FLET : SKLV :

CDEV : BKIN : MOUT : ADMN : SLIB :

EQMT : CTRN : ISCH : ANLV :

DOTS MAINTENANCE PROGRAM: Filename = MPGM3
Filetype = PLI

PROGRAMMING LANGUAGE : PL/1

NCSS EXECUTIVES : REFORM

PROGRAM FUNCTION

A card-image file is produced as a result of the execution of a RAMIS dump function. 80-byte output records are produced regardless of the length of the RAMIS file internal record. A DUMP function directed at each of the DOTS files results in the following output:

Facilities file: One card output per internal record

Instructor file: Two cards required per internal record

Course file: Three cards per internal record are required

Use of the dump output to control subsequent RAMIS maintenance functions requires that the multiple card images be combined into single records. Program MPGM3 combines single 80-byte input records into multiples of two (160-byte output records) or multiples of three (240-byte output records).

PROGRAM MPGM3 INPUT/OUTPUT

MPGM3 requires a primary card-input data set. Input as parameter input from executive = REFORM is a single digit, value = 2 or 3. This parameter input directs blocking the output at two (160 bytes) or three (240 bytes).

Listing of Filename = MPGM3, Filetype = PLI

```
MPGM3:  /* COMBINE 2/MORE CARDS INTO A SINGLE LRECL - 20/MAY/76 */
        PROC (PARMS) OPTIONS (MAIN) ;

DECLARE PARMS CHAR (100) VARYING,
        NEOF FIXED BIN (31,0) STATIC,
        IHESARC ENTRY ( FIXED BINARY (31,0)- ) ;

DECLARE CARDS (240) CHAR (1) STATIC,
        CARD1 CHAR (80) BASED (PC1),
        CARD2 CHAR (80) BASED (PC2),
        CARD3 CHAR (80) BASED (PC3),
        BUF2 CHAR (160) BASED (PB),
        BUF3 CHAR (240) BASED (PB) ;
```

Listing of Filename = MPGM3, Filetype = PLI (Cont'd)

```
STARTERHUPP:
        GET STRING (PARMS) EDIT (NTOT) ( F(4) ) ;

        PB = ADDR (CARDS(1)) ;
        PC1 = ADDR (CARDS(1)) ;
        PC2 = ADDR (CARDS(81)) ;
        PC3 = ADDR (CARDS(161)) ;
        NR, NC = 0 ;

        OPEN FILE (DUMP) RECORD INPUT,
             FILE (DUMP2) RECORD OUTPUT ;
        ON ENDFILE (DUMP) GO TO EOJ ;

R1:     NEOF = 0 ;
        READ FILE (DUMP) INTO (CARD1) ;
             NC = NC + 1 ;
        NEOF = 4 ;
        READ FILE (DUMP) INTO (CARD2) ;
             NC = NC + 1 ;

        IF ( NTOT = 2 ) THEN GO TO W2 ;

        READ FILE (DUMP) INTO (CARD3) ;
             NC = NC + 1 ;

W3:         WRITE FILE (DUMP2) FROM (BUF3) ;
W3A:        NR = NR + 1 ;
                GO TO R1 ;

W2:     WRITE FILE (DUMP2) FROM (BUF2) ;
                GO TO W3A ;

EOJ:    DISPLAY ( ' ' ) ;
        DISPLAY ( NC || ' : TOTAL INPUT DUMP RECORDS' ) ;
        DISPLAY ( NR || ' : TOTAL OUTPUT COMBINATION RECORDS' ) ;

        CLOSE FILE (DUMP), FILE (DUMP2) ;

        CALL IHESARC ( NEOF ) ;

             END MPGM3 ;
```

DOTS MAINTENANCE PROGRAM : Filename = PCOURSE
Filetype = PLI

PROGRAMMING LANGUAGE : PL/1

NCSS EXECUTIVES : PCOURSE
PCB

PROGRAM FUNCTION

Program PCOURSE produces a formatted printer output of the contents of the DOTS Course file. The complete contents (all data fields) of all records of the Course file are printed.

PROGRAM PCOURSE INPUT/OUTPUT

Input to PCOURSE is the reformatted dump file COURSE/DUMP2 (filename/filetype). This file contains 240-byte records which resulted from a previous RAMIS dump. See the discourse for maintenance program = MPGM3 in this section of this document.

Output from PCOURSE is shown in the following illustration. The complete dump file is printed on a single run of PCOURSE. The NCSS executive PCOURSE forces a resort of the dump file into prime/secondary sort order by School Code/Course CDP number before printing.

Sample of Program PCOURSE Output

```
        DOTS COURSE FILE LISTING (FILENAME = COURSE)                    PAGE :   16
                ON 80/09/?n AT 1?:?/

CUP   SCH      LFLG          CNS              CTYPF  PNEC  SUIC   NOTE      CCNN          PRTY  FPRTY                    SIR/ID       CNRS/PQUAL  TY/ASN
----  ----  ----------  ----------------  -----  ----  -----  ------  ----------  ----  -----                    ----------   ----------  -----
                                                                                                                5044647050   100         1

V342  SINT  U= JA-0950  BASIC INTEL HES     F2          0300A 770106  FITCLANT     7                                0            0         0
      CONV1 :    6     CONV2 :    6     CONV3 :    6     MAXCAP :  20   PFAIL : 5.000   MSETB :  .000          10           5         0
      LEN1  :   12     LEN2  :   12     LEN3  :   12     MINCAP :   5   MADIS :  .020   TIMSTR :  .5           25           50        0
      CAP1  :   20     CAP2  :   20     CAP3  :   20     PCAP   :  20   MOSHOU :  .050  UDEM   :  0 0   1303452210   100         1
      HCAP1 :    0     HCAP2 :    0     HCAP3 :    0     ECAP   :  20   SETB  :  .000   OPT1   :       2774280748   100         1
      DMD1  :   74     DMD2  :   74     DMD3  :   74     SCAP   :  24   BKLG  :    0    OPT2   :       3234647393   100         1
      HDMD1 :    0     HDMD2 :    0     HDMD3 :    0     UCAP   :  84   PADB  :    0    OPT3   :       3354047714   100         1
                                                                                                        4414426831   100         1
                                                                                                        4579230645   100         1
                                                                                                        5390463563   100         1
                                                                                                        5415000004   100         1
                                                                                                        5044647050   100         1

V343  SINT  U= JA-0955  AIM INTEL HES       F2          0300A 770106  FITCLANT     7                                0            0         0
      CONV1 :    6     CONV2 :    6     CONV3 :    6     MAXCAP :  20   PFAIL : 4.000   MSETB :  .000          25           25        0
      LEN1  :   12     LEN2  :   12     LEN3  :   12     MINCAP :   5   MADIS :  .000   TIMSTR :  .5   1523000009   100         1
      CAP1  :   20     CAP2  :   20     CAP3  :   20     PCAP   :  20   MOSHOU :  .050  UDEM   :  0 0   2774280748   100         1
      HCAP1 :    0     HCAP2 :    0     HCAP3 :    0     ECAP   :  20   SETB  :  .000   OPT1   :       3304047714   100         1
      DMD1  :   56     DMD2  :   56     DMD3  :   56     SCAP   :   6   BKLG  :    0    OPT2   :       4414426831   100         1
      HDMD1 :    0     HDMD2 :    0     HDMD3 :    0     UCAP   :  64   PADB  :    0    OPT3   :       5390463563   100         1

V344  SINT  U= JA-0954  IMAGE INTERP HES     F2          0300A 770106  FITCLANT     7                                0            0         0
      CONV1 :    5     CONV2 :    5     CONV3 :    5     MAXCAP :  24   PFAIL : 3.000   MSETB :  .000          25           40        0
      LEN1  :   12     LEN2  :   12     LEN3  :   12     MINCAP :   5   MADIS :  .000   TIMSTR :  2.9  3064665540   100         1
      CAP1  :   20     CAP2  :   20     CAP3  :   20     PCAP   :   0   MOSHOU :  .050  UDEM   :  0 0   3234647393   100         1
      HCAP1 :    0     HCAP2 :    0     HCAP3 :    0     ECAP   :   0   SETB  :  .000   OPT1   :       4414426831   100         1
      DMD1  :   85     DMD2  :   85     DMD3  :   85     SCAP   :   0   BKLG  :    0    OPT2   :       5390463563   100         1
      HDMD1 :    0     HDMD2 :    0     HDMD3 :    0     UCAP   :  84   PADB  :    0    OPT3   :

V345  SINT  K= JA-5014  NAV INT FUNN OFF    C2          0300A 770331  FITCPAC      0                                0            0         0
      CONV1 :    1     CONV2 :    1     CONV3 :    1     MAXCAP :  14   PFAIL :  .000   MSETB :  .000           3            7         0
      LEN1  :   54     LEN2  :   54     LEN3  :   54     MINCAP :   5   MADIS :  .000   TIMSTR :  .4           4            12        0
      CAP1  :   14     CAP2  :   14     CAP3  :   14     PCAP   :  14   MOSHOU :  .100  UDEM   :  0 0           6            105       0
      HCAP1 :    0     HCAP2 :    0     HCAP3 :    0     ECAP   :  14   SETB  :  .000   OPT1   :       1303452210   100         1
      DMD1  :    8     DMD2  :    8     DMD3  :    8     SCAP   :  14   BKLG  :    0    OPT2   :       1523000009   100         1
      HDMD1 :    0     HDMD2 :    0     HDMD3 :    0     UCAP   :  24   PADB  :    0    OPT3   :       2774280748   100         1
                                                                                                        3064665540   100         1
                                                                                                        3234647393   100         1
```

Listing of Filename = PCOURSE, Filetype = PLI

```
PCOURSE: /* PRINT COURSE FILE FROM DUMP OUTPUT - 7/JUL/76 */
        PROC OPTIONS (MAIN) ;

DECLARE 1 COURSE  BASED (P1),
          2 CDP    CHAR (4),
          2 SCH    CHAR (4),
          2 DEL    CHAR (1),
          2 CTLG   CHAR (10),
          2 CRS    CHAR (16),
          2 CONV1  CHAR (3),
          2 LEN1   CHAR (4),
          2 CAP1   CHAR (3),
          2 BCAP1  CHAR (3),
          2 DMD1   CHAR (5),
          2 BDMD1  CHAR (5),
          2 OFST   CHAR (2),
          2 PFAIL  CHAR (5),
          2 NADIS  CHAR (5),
          2 NOSHOW      CHAR (5),
          2 SETB   CHAR (5),
             /* END OF FIRST DUMP CARD IMAGE: 80 BYTES EXACTLY */
          2 BKLG   CHAR (4),
          2 MAXCAP      CHAR (4),
          2 CONV2  CHAR (3),
          2 LEN2   CHAR (4),
          2 CAP2   CHAR (3),
          2 BCAP2  CHAR (3),
          2 DMD2   CHAR (5),
          2 BDMD2  CHAR (5),
          2 CHWK2  CHAR (3),
          2 CONV3  CHAR (3),
          2 LEN3   CHAR (4),
          2 CAP3   CHAR (3),
          2 BCAP3  CHAR (3),
          2 DMD3   CHAR (5),
          2 BDMD3  CHAR (5),
          2 CHWK3  CHAR (3),
          2 MINCAP      CHAR (4),
          2 AFAIL  CHAR (5),
          2 PRTY   CHAR (2),
          2 CTYPE  CHAR (2),
          2 PNEC   CHAR (4),
          2 UCAP   CHAR (3),
             /* END OF SECOND CARD: 80 BYTES EXACTLY */
          2 FPRTY  CHAR (2),
          2 UDEM   CHAR (3),
          2 RDTE   CHAR (6),
          2 HSETB  CHAR (5),
          2 TINSTR      CHAR (4),
          2 SUIC   CHAR (5),
          2 CCMM   CHAR (10),
          2 OPT1   CHAR (6),
          2 OPT2   CHAR (6),
          2 OPT3   CHAR (6),
          2 ECAP   CHAR (3), V-190
```

Listing of Filename = PCOURSE, Filetype = PLI (Cont'd)

```
            2 SCAP   CHAR (3),
            2 PCAP   CHAR (3),
            2 PAOB   CHAR (4),
            2 F1     CHAR (9),
            2 F2     CHAR (4),
            2 F3     CHAR (1) ;

 DECLARE RC2  CHAR (240) STATIC,
         OLDSCH     CHAR (4) STATIC INIT ( ' ' ),
         CDP2 CHAR (4) DEF RC2,
         FF1  CHAR (9) DEF RC2 POS (227),
         FF2  CHAR (4) DEF RC2 POS (236),
         FF3  CHAR (1) DEF RC2 POS (240),
         MAKE CHAR (21) STATIC INIT ( ' ' ),
         M1   CHAR (9) DEF MAKE POS (1),
         M2   CHAR (4) DEF MAKE POS (13),
         M3   CHAR (1) DEF MAKE POS (20),
         SAVE (100) CHAR (21) STATIC ;

 DECLARE RCOURSE   CHAR (240) STATIC,
         LMAX STATIC INIT ( 58 ),
         REOF BIT (1) STATIC INIT ( '0'B ),
         DGET CHAR (6) STATIC,
         YY   CHAR (2) DEF DGET,
         MM   CHAR (2) DEF DGET POS (3),
         DD   CHAR (2) DEF DGET POS (5) ;

 DECLARE T1   CHAR (44) STATIC INIT (
              'DOTS COURSE FILE LISTING (FILENAME = COURSE)' ),
         T2   CHAR (19) STATIC INIT (
              'ON XX/XX/XX AT XXXX' ),
         DTE  CHAR (8) DEF T2 POS (4),
         TME  CHAR (4) DEF T2 POS (16),
         T3   CHAR (132) STATIC INIT (
 'CDP   SCH      CTLG            CRS          CTYPE  PNEC  SUIC   RDTE
   CCMM     PRTY  FPRTY                       SIR/     CHRS/  TY/' ),
         T4   CHAR (132) STATIC INIT (
 '----  ----  ----------  ---------------  -----  ----  ----- ------
----------  ----  -----                    ---------  -----  ---' ) ;

 START:  OPEN FILE (DUMP) RECORD INPUT,
              FILE (SYSPRINT) STREAM PRINT
                   LINESIZE (132) PAGESIZE (58) ;

         ON ENDFILE (DUMP) GO TO EOJ ;

         ON ENDPAGE (SYSPRINT)
              BEGIN ;
              NPAGE = NPAGE + 1 ;

              PUT EDIT ( T1, 'PAGE : ', NPAGE, T2, T3,
                   'ID      PQUAL  ASN', T4 )
                   ( PAGE, LINE(3),
                     COL(10), A(44), COL(70), A, F(4),
                     SKIP(1), COL(20), A(19),
```

Listing of Filename = PCOURSE, Filetype = PLI (Cont'd)

```
                    SKIP(2), A(132),
                  SKIP(1), COL(115), A,
                  SKIP(1), A(132) ) ;

            PUT SKIP (1) ;
            END ;

        P1 = ADDR (RCOURSE) ;
        DGET = DATE ;
        DTE = MM || '/' || DD || '/' || YY ;
        TME = SUBSTR ( TIME, 1, 4 ) ;
        NC, NPAGE = 0 ;
        LMAX = LMAX - 9 ;
        SIGNAL ENDPAGE (SYSPRINT) ;

R1:     READ FILE (DUMP) INTO (RCOURSE) ;
        NC = NC + 1 ;
            OLDSCH = SCH ;

STEPA:  NN = 1 ;
        M1 = F1 ;  M2 = F2 ;  M3 = F3 ;
            SAVE(1) = MAKE ;

R2:     READ FILE (DUMP) INTO (RC2) ;
        NC = NC + 1 ;

        IF ( CDP2 ¬= CDP ) THEN GO TO STEPX ;

            NN = NN + 1 ;
            M1 = FF1 ;  M2 = FF2 ;  M3 = FF3 ;
            SAVE(NN) = MAKE ;
                GO TO R2 ;

STEPX:  IF ( NN >= 9 ) THEN GO TO STEPP ;

            NN = NN + 1 ;
            SAVE(NN) = ' ' ;
                GO TO STEPX ;

STEPP:  IF ( LINENO(SYSPRINT) > LMAX )
            THEN SIGNAL ENDPAGE (SYSPRINT) ;

L12:    PUT EDIT ( CDP, SCH, CTLG, CRS, CTYPE, PNEC,
            SUIC, RDTE, CCMM, PRTY, FPRTY, SAVE(1), SAVE(2) )
            ( SKIP(3), A(4), X(2), A(4), X(2), A(10),
              X(2), A(16), X(3), A(2), X(4), A(4), X(2), A(5),
              X(1), A(6), X(2), A(10), X(3), A(2),
              X(4), A(2), X(21), A(21),
              SKIP(1), COL(112), A(21) ) ;

L3:     PUT EDIT ( '    CONV1 :   ', CONV1,
            '    CONV2 :   ', CONV2,
            '    CONV3 :   ', CONV3,
            '    MAXCAP : ', MAXCAP,
            '    PFAIL  : ', PFAIL,
```

Listing of Filename = PCOURSE, Filetype = PLI (Cont'd)

```
                '    HSETH  :  ', HSETH, SAVE(3) )
               ( SKIP(1), X(2), 3 ( A(14), A(3) ), A, A(4),
                 A, A(5), A, A(5), X(4), A(21) ) ;

L4:      PUT EDIT ( '    LEN1  :  ', LEN1,
                '    LEN2  :  ', LEN2,
                '    LEN3  :  ', LEN3,
                '    MINCAP :  ', MINCAP,
                '    NADIS  :  ', NADIS,
                '    TINSTH :   ', TINSTH, SAVE(4) )
               ( SKIP(1), X(2), 4 ( A(13), A(4) ), A, A(5),
                 A, A(4), X(4), A(21) ) ;

L5:      PUT EDIT ( '    CAP1  :   ', CAP1,
                '    CAP2  :   ', CAP2,
                '    CAP3  :   ', CAP3,
                '    PCAP  :   ', PCAP,
                '    NOSHOW :  ', NOSHOW,
                '    UDEM   :    ', UDEM, SAVE(5) )
               ( SKIP(1), X(2), 4 ( A(14), A(3) ), A, A(5),
                 A, A(3), X(4), A(21) ) ;

L6:      PUT EDIT ( '    BCAP1 :   ', BCAP1,
                '    BCAP2 :   ', BCAP2,
                '    BCAP3 :   ', BCAP3,
                '    ECAP  :   ', ECAP,
                '    SETH   :  ', SETH,
                '    OPT1   :  ', OPT1, SAVE(6) )
               ( SKIP(1), X(2), 4 ( A(14), A(3) ), A, A(5),
                A, A(6), X(4), A(21) ) ;

L7:      PUT EDIT ( '    DMD1  :  ', DMD1,
                '    DMD2  :  ', DMD2,
                '    DMD3  :  ', DMD3,
                '    SCAP   :  ', SCAP,
                '    BKLG   :  ', BKLG,
                '    OPT2   :  ', OPT2, SAVE(7) )
               ( SKIP(1), X(2), 3 ( A(12), A(5) ), A, A(3),
                 A, A(4), A, A(6), X(4), A(21) ) ;

L8:      PUT EDIT ( '    BDMD1 :  ', BDMD1,
                '    BDMD2 :  ', BDMD2,
                '    BDMD3 :  ', BDMD3,
                '    UCAP   :  ', UCAP,
                '    PAOB   :  ', PAOB,
                '    OPT3   :  ', OPT3, SAVE(8) )
               ( SKIP(1), X(2), 3 ( A(12), A(5) ), A, A(3),
                 A, A(4), A, A(6), X(4), A(21) ) ;

AFTERPRINT:
         IF ( NN <= 8 ) THEN GO TO STEPY ;

               N = 9 ;
STEPN:         PUT EDIT ( SAVE(N) ) ( SKIP(1), COL(112), A(21) ) ;
               N = N + 1 ;
```

Listing of Filename = PCOURSE, Filetype = PLI (Cont'd)

```
          IF ( N <= NN ) THEN GO TO STEPN ;

STEPY:    IF ( REOF ) THEN GO TO WINDUP ;

          RCOURSE = RC2 ;
          IF ( SCH = OLDSCH ) THEN GO TO STEPA ;
               OLDSCH = SCH ;
               SIGNAL ENDPAGE (SYSPRINT) ;
               GO TO STEPA ;

EOJ:      REOF = '1'H ;
          GO TO STEPX ;

WINDUP:
          SIGNAL ENDPAGE (SYSPRINT) ;
          PUT LIST ( '*** NOW AT EOF ***' ) SKIP (3) ;
          PUT LIST ( '*** ' || NC || ' : DUMP RECORDS PROCESSED' )
               SKIP (1) ;

          DISPLAY ( NC || 'DUMP RECORDS PROCESSED' ) ;

          CLOSE FILE (DUMP), FILE (SYSPRINT) ;

               END PCOURSE ;
```

DOTS MAINTENANCE PROGRAM : Filename = PFACIL
Filetype = PLI

PROGRAMMING LANGUAGE : PL/1

NCSS EXECUTIVES : PFACIL
PFCB

PROGRAM FUNCTION

This program serves to provide a printer output of the contents of the complete DOTS Facilities file. The NCSS card-image file FACIL/DUMP is used as an input to this program.

PROGRAM PFACIL INPUT/OUTPUT

The input for PFACIL is taken from the dump file FACIL/DUMP (filename/filetype). This input is in card-image (80-byte) format; the file was created by a previous run of a RAMIS dump function.

The NCSS executive PFACIL resorts the dump file into School Code and CDP Number order. The original dump file is in prime sort CDP Number order.

Output from PFACIL is shown in the following example. The complete dump file is printed on a single run of PFACIL.

Listing of Filename = PFACIL, Filetype = PLI

```
PFACIL: /* PRINT FACIL FILE FROM DUMP OUTPUT - 7/JUL/76 */
        PROC OPTIONS (MAIN) ;

DECLARE 1 FACIL    BASED (P1),
          2 CDP    CHAR (4),
          2 SCH    CHAR (4),
          2 BLDG   CHAR (4),
          2 RM     CHAR (4),
          2 RMCAP    CHAR (3),
          2 RMTP     CHAR (1),
          2 REQHRS   CHAR (6),
          2 AVHRS    CHAR (5) ;

DECLARE RFACIL     CHAR (80) STATIC,
        OLDSCH     CHAR (4) STATIC INIT ( 'XXXX' ),
        OLDCDP     CHAR (4) STATIC INIT ( ' ' ),
        DGET CHAR (6) STATIC,
        YY   CHAR (2) DEF DGET,
        MM   CHAR (2) DEF DGET POS (3),
        DD   CHAR (2) DEF DGET POS (5) ;
```

Listing of Filename = PFACIL, Filetype = PLI (Cont'd)

```
DECLARE T1    CHAR (47) STATIC INIT (
              'DOTS FACILITIES FILE LISTING (FILENAME = FACIL)' ),
        T2    CHAR (19) STATIC INIT (
              'ON XX/XX/XX AT XXXX' ),
        DTE   CHAR (8) DEF T2 POS (4),
        TME   CHAR (4) DEF T2 POS (16),
        T3    CHAR (50) STATIC INIT (
              'CDP   SCH   BLDG    RM  RMCAP  RMTP  REQHRS  AVHRS' ),
        T4    CHAR (50) STATIC INIT (
              '----  ----  -----  ---  -----  ----  ------  -----' ) ;

START:  OPEN FILE (DUMP) RECORD INPUT,
             FILE (SYSPRINT) STREAM PRINT
                  LINESIZE (132) PAGESIZE (58) ;

        ON ENDFILE (DUMP) GO TO EOJ ;

        ON ENDPAGE (SYSPRINT)
             BEGIN ;
             NPAGE = NPAGE + 1 ;

             PUT EDIT ( T1, 'PAGE : ', NPAGE, T2, T3, T4 )
                  ( PAGE, LINE(3),
                    COL(10), A(47), COL(70), A, F(4),
                    SKIP(1), COL(20), A(19),
                    SKIP(2), A(50), SKIP(1), A(50) ) ;

             PUT SKIP (1) ;
             END ;

        P1 = ADDR (RFACIL) ;
        DGET = DATE ;
        DTE = MM || '/' || DD || '/' || YY ;
        TME = SUBSTR ( TIME, 1, 4 ) ;
        NC, NPAGE = 0 ;

R1:     READ FILE (DUMP) INTO (RFACIL) ;
        NC = NC + 1 ;

STEP2:  IF ( SCH = OLDSCH ) THEN GO TO STEP3 ;
             OLDSCH = SCH ;
             SIGNAL ENDPAGE (SYSPRINT) ;

STEP3:  IF ( CDP = OLDCDP ) THEN GO TO STEP1 ;
             PUT SKIP (1) ;
             OLDCDP = CDP ;
```

Listing of Filename = PFACIL, Filetype = PLI (Cont'd)

```
STEP1:  PUT EDIT ( FACIL )
              ( SKIP(1), A(4), X(2), A(4), X(2), A(4),
                X(2), A(4), X(3), A(3), X(4), A(1),
                X(4), A(6), X(2), A(5) ) ;

                  GO TO R1 ;

EOJ:    SIGNAL ENDPAGE (SYSPRINT) ;
        PUT LIST ( '*** NOW AT EOF ***' ) SKIP (3) ;
        PUT LIST ( '*** ' || NC || ' : DUMP RECORDS PROCESSED' )
             SKIP (1) ;

        DISPLAY ( NC || ' DUMP RECORDS PROCESSED' ) ;

        CLOSE FILE (DUMP), FILE (SYSPRINT) ;

             END PFACIL ;
```

Sample of Program PFACIL Output

DUTS FACILITIES FILE LISTING (FILENAME = FACIL)
ON 08/07/7- AT 1746

CDP	SCH	BLDG	RM	RMCAP	RMTP	REQHRS	AVHRS
1130	430	678	116	15	3	7.0	1750
1130	430	678	118	15	2	30.0	1750
1130	430	678	121	10	2	20.0	1750
1130	430	678	137	15	2	57.0	1750
1130	430	678	239	15	1	138.0	1750
1262	430	678	121	0	2	10.0	1750
1262	430	678	221	0	1	46.0	1750
205K	430	678	116	10	2	16.0	1750
205K	430	678	221	20	1	19.0	1750
2625	430	678	112	25	1	12.0	1750
2625	430	678	118	6	2	23.0	1750
323G	430	678	149	20	2	48.0	1750
323G	430	678	209	40	1	148.0	1750
506T	430	678	116	8	2	9.0	1750
506T	430	678	221	30	1	19.0	1750
506V	430	678	116	12	2	29.0	1750
506V	430	678	221	30	1	20.0	1750
506Y	430	678	221	20	1	8.0	1750
506Z	430	678	112	25	1	10.0	1750
506Z	430	678	134	16	4	11.0	625
507C	430	678	118	18	1	13.0	1750
507C	430	678	134	16	4	22.0	625
507F	430	678	118	16	3	21.0	1750
507F	430	678	229	40	1	14.0	1750
540Z	430	678	221	20	1	20.5	1750
6025	430	678	108	20	3	15.0	1750
6025	430	678	112	25	1	10.0	1750
6025	430	678	116	15	3	15.5	1750
6025	430	678	118	15	2	27.0	1750
6025	430	678	121	20	2	19.0	1750
6025	430	678	125	20	2	23.0	1750
6025	430	678	139	15	2	13.0	1750
6025	430	678	149	20	3	53.0	1750
6025	430	678	233	20	1	15.0	1750
6025	430	678	235	20	1	102.0	1750

DOTS MAINTENANCE PROGRAM : Filename = PINST
Filetype = PLI

PROGRAMMING LANGUAGE : PL/1

NCSS EXECUTIVES : PINST
PICB

PROGRAM FUNCTION

A formatted printer output of the contents of the DOTS Instructor file is produced by program PINST. The complete contents (all data fields in all records) are printed in a single run of PINST.

PROGRAM PINST INPUT/OUTPUT

Input to this program is the reformatted dump file INST/DUMP2 (filename/filetype). This file contains 160-byte records which resulted from a previous RAMIS dump. The reformatting operation is discussed in the section covering program = MPGM3 in this document.

Output from program PINST is shown in the following sample. The complete dump file is printed on a single run of PINST. The NCSS executive PINST sorts the dump file into School Code/Instructor ID order before printing.

Listing of Filename = PINST, Filetype = PLI

```
PINST: /* PRINT INST FILE FROM DUMP OUTPUT - 7/JUL/76 */
        PROC OPTIONS (MAIN) ;

DECLARE 1 INST     BASED (P1),
          2 ID     CHAR (9),
          2 NME    CHAR (16),
          2 IN     CHAR (2),
          2 RATE   CHAR (5),
          ( 2 SCH,  2 REP,  2 ROT,  2 PNEC,
            2 SNEC,  2 NEC3,  2 NEC4,  2 NEC5,
            2 CONT,  2 CDEV,  2 EQMT,  2 PREP,
            2 BKIN,  2 CTRN,  2 FTRN,  2 MDUT,
            2 ISCH,  2 FLET,  2 ADMN,  2 ANLV,
            2 SKLV,  2 SLIB )   CHAR (4) ;

DECLARE RINST      CHAR (160) STATIC,
        OLDSCH     CHAR (4) STATIC INIT ( 'XX X' ),
        LMAX STATIC INIT ( 58 ),
        DGET CHAR (6) STATIC,
        YY   CHAR (2) DEF DGET,
        MM   CHAR (2) DEF DGET POS (3),
        DD   CHAR (2) DEF DGET POS (5) ;
```

Listing of Filename = PINST, Filetype = PLI (Cont'd)

```
DECLARE T1    CHAR (46) STATIC INIT (
              'DOTS INSTRUCTOR FILE LISTING (FILENAME = INST)' ),
        T2    CHAR (19) STATIC INIT (
              'ON XX/XX/XX AT XXXX' ),
        DTE   CHAR (8) DEF T2 POS (4),
        TME   CHAR (4) DEF T2 POS (16),
        T3    CHAR (86) STATIC INIT (
              '   ID       RATE   IN          NME          SCH   REP   ROT
PNEC  SNEC    NEC3  NEC4  NEC5' ),
        T4    CHAR (86) STATIC INIT (
              '---------  -----  --  ---------------  ----  ----  ----
 ----  ----   ----  ----  ----' ) ;

START:  OPEN FILE (DUMP) RECORD INPUT,
             FILE (SYSPRINT) STREAM PRINT
                  LINESIZE (132) PAGESIZE (58) ;

        ON ENDFILE (DUMP) GO TO EOJ ;

        ON ENDPAGE (SYSPRINT)
             BEGIN ;
             NPAGE = NPAGE + 1 ;

             PUT EDIT ( T1, 'PAGE : ', NPAGE, T2, T3, T4 )
                  ( PAGE, LINE(3),
                    COL(10), A(46), COL(70), A, F(4),
                    SKIP(1), COL(20), A(19),
                    SKIP(2), A(86), SKIP(1), A(86) ) ;

             PUT SKIP (1) ;
             END ;

        P1 = ADDR (RINST) ;
        DGET = DATE ;
        DTE = MM || '/' || DD || '/' || YY ;
        TME = SUBSTR ( TIME, 1, 4 ) ;
        NC, NPAGE = 0 ;
        LMAX = LMAX - 9 ;

R1:     READ FILE (DUMP) INTO (RINST) ;
        NC = NC + 1 ;

        IF ( SCH = OLDSCH ) THEN GO TO STEP1 ;
             OLDSCH = SCH ;
             SIGNAL ENDPAGE (SYSPRINT) ;
```

Listing of Filename = PINST, Filetype = PLI (Cont'd)

```
STEP1: IF ( LINENO(SYSPRINT) > LMAX )
          THEN SIGNAL ENDPAGE (SYSPRINT) ;

       PUT EDIT ( ID, RATE, IN, NME, SCH, REP, ROT,
            PNEC, SNEC, NEC3, NEC4, NEC5 )
            ( SKIP(3), A(9), X(2), A(5), X(2), A(2),
              X(2), A(16), 8 ( X(2), A(4) ) ) ;

       PUT EDIT ( 'CONT : ', CONT, '    PREP : ', PREP,
            '    FTRN : ', FTRN, '    FLET : ', FLET,
            '    SKLV : ', SKLV )
            ( SKIP(2), COL(12), A, A(4), 4 ( A(11), A(4) ) ) ;

       PUT EDIT ( 'CDEV : ', CDEV, '    BKIN : ', BKIN,
            '    MOUT : ', MOUT, '    ADMN : ', ADMN,
            '    SLIB : ', SLIB )
            ( SKIP(1), COL(12), A, A(4), 4 ( A(11), A(4) ) ) ;

       PUT EDIT ( 'EQMT : ', EQMT, '    CTRN : ', CTRN,
            '    ISCH : ', ISCH, '    ANLV : ', ANLV )
            ( SKIP(1), COL(12), A, A(4), 3 ( A(11), A(4) ) ) ;

            GO TO R1 ;

EOJ:   SIGNAL ENDPAGE (SYSPRINT) ;
       PUT LIST ( '*** NOW AT EOF ***' ) SKIP (3) ;
       PUT LIST ( '*** ' || NC || ' : DUMP RECORDS PROCESSED' )
            SKIP (1) ;

       DISPLAY ( NC || ' DUMP RECORDS PROCESSED' ) ;

       CLOSE FILE (DUMP), FILE (SYSPRINT) ;

            END PINST ;
```

Sample of Program PINST Output

036235

```
        DOTS INSTRUCTOR FILE LISTING (FILENAME = INST)                    PAGE :   45
                    ON 08/07/75 AT 1746

    ID        RATE  IN        NME          SCH   REP   ROT   PNEC  SNEC  NEC3  NEC4  NEC5
 ---------    -----  --  ----------------  ----  ----  ----  ----  ----  ----  ----  ----

 473626394    ARH1   DB  HANSEN            6E    7605  7906

              CONT :  600    PREP :  100    FTRN :    0    FLET :    0    SKLV :    0
              CDEV :   15    BKIN :  750    MDUT :  120    ADMN :   50    SLIB :    0
              EQMT :  140    CTRN :    0    ISCH :  160    ANLV :  280

 480340003    HTC    CE  PUGH              6E    7404  7704        9508

              CONT :  776    PREP :   48    FTRN :    0    FLET :    0    SKLV :    8
              CDEV :   46    BKIN :   30    MDUT :  190    ADMN :  250    SLIB :  336
              EQMT :   36    CTRN :    0    ISCH :    0    ANLV :  360

 481708667    HT2    R   JEFFERY           6E    7509  7809        9502

              CONT :  500    PREP :  250    FTRN :    0    FLET :  100    SKLV :    0
              CDEV :   50    BKIN :   20    MDUT :  100    ADMN :   50    SLIB :   30
              EQMT :  100    CTRN :   60    ISCH :  200    ANLV :  620

 490462714    MMC    JH  EARNEST           6E    7509  7809  4294

              CONT :  721    PREP :  125    FTRN :    0    FLET :   75    SKLV :    0
              CDEV :   45    BKIN :   75    MDUT :  152    ADMN :  175    SLIB :  175
              EQMT :   75    CTRN :    0    ISCH :  150    ANLV :  312

 492423681    ABH1   HL  STINSON           6E    7402  7702        9502

              CONT :  204    PREP :  208    FTRN :   32    FLET :  100    SKLV :    0
              CDEV :    0    BKIN :  100    MDUT :  852    ADMN :  100    SLIB :    0
              EQMT :  104    CTRN :    0    ISCH :  240    ANLV :  240

 503622306    HT1    D   SHOUN             6E    7403  7703  4951  9502

              CONT :  482    PREP :  200    FTRN :   66    FLET :   50    SKLV :    0
              CDEV :   36    BKIN :   36    MDUT :  140    ADMN :    0    SLIB :   30
              EQMT :   60    CTRN :  120    ISCH :  200    ANLV :  580
```

DOTS MAINTENANCE PROGRAM : Filename = SUPRPGM
Filetype = PLI

PROGRAMMING LANGUAGE : PL/1

NCSS EXECUTIVES : SUPER
SUPERX

PROGRAM FUNCTION

This program is referred to in Figure II-1 as the Input Transaction Processing Program. SUPRPGM input is the DOTS system maintenance transactions as transmitted from the user site via the WANG computer system. These transactions are used directly as transmitted or reformatted as required to effect the maintenance functions.

PROGRAM SUPRPGM INPUT/OUTPUT

Input to this program is covered in detail in separate sections. Each transaction type is covered individually both in input format and output resulting from SUPRPGM.

Listing of Filename = SUPRPGM, Filetype = PLI

```
SUPRPGM: /* SUPER TRANSACTION PROCESSOR: MON/19/APR/76 */
         PROC OPTIONS (MAIN) ;

         /* REVISIONS :
         WED/21/JUL/76 - DEBUG FIXES TO ELIMINATE 'ID' AND 'SIR' NAMES
                         FROM COURSE FILE PROCESSING.  SHOULD BE 'F1'.

         FRI/13/AUG/76 - FIXED TYPES 2, 14, 19, 25 TRANSACTIONS
                         TO ALLOW BLDG AND RM FIELDS OF FACIL LENGTH
                         CHANGE TO A(4) EACH.
                       - ALSO REMOVED SLASH INSERTION IN GETFBL
                         SUBROUTINE.
         */

DECLARE  BUF      CHAR (80) STATIC,
         TRAN     CHAR (80) STATIC,
         BUFA (80) CHAR (1) BASED (PB),
         TRANSA (80) CHAR (1) BASED (PT),
         CC1      CHAR (1) DEF TRAN,
         F64      CHAR (64) DEF TRAN,
         F40      CHAR (40) DEF BUF,
         TNF      CHAR (2) DEF TRAN,
         TRAN64     CHAR (64) DEF TRAN,
         NUMB     PICTURE '9999',
         TSEQ     PICTURE '9999' DEF TRAN POS (61) ;

DECLARE  ( SRET1, SRET2, READR, REOF, RGEN, RBUST, SRET3,
           COM678, VECTOR(27) ) LABEL ;

DECLARE  ( COPEN, AOPEN, BOPEN, FLAGEOF ) BIT (1) STATIC,
         PFLAG BIT (1) STATIC INIT ( '1'B ) ;

DECLARE  BLANK    CHAR (4) STATIC INIT ( ' ' ),
         NN        FIXED BIN (31,0) STATIC INIT ( 0 ),
         ERMSG    CHAR (13) STATIC INIT ( '???(XXXXX)???' ),
         NNER     PICTURE '99999' DEF ERMSG POS (5),
         DERM     CHAR (21) STATIC INIT (
              'ERROR ONCODE WAS XXXX' ),
         NONC     PICTURE '9999' DEF DERM POS (18),
         DERM2    CHAR (23) STATIC INIT (
              'CONVERSION ROUTE WAS XX' ),
         NNCER    PICTURE '99' DEF DERM2 POS (22) ;

DECLARE  ( NUM2, NUM2A ) PICTURE 'Z9' STATIC,
         DS1320   CHAR (8) STATIC INIT ( ' ' ),
         DSOPEN   CHAR (8) STATIC INIT ( ' ' ),
         FILLF  CHAR (1) STATIC INIT ( ',' ),
         NTRAN    PICTURE 'ZZZ9' STATIC,
         JMAX     STATIC INIT ( 10 ),
         IHESARC ENTRY ( FIXED BINARY (31,0) ) ;

DECLARE  SS1320 ( 13:20, 10 ) FIXED BIN STATIC INIT (
              7, 12, 13, 15, (6)0,
              0, 0, (8)0,
              7, 10, (8)0,
```

Listing of Filename = SUPRPGM, Filetype = PLI (Cont'd)

```
                7, 10, 11, 14, (6)0,
                7, 10, 11, 13, 14, 16, (4)0,
                7, 12, 13, 15, 16, 21, 22, 24, (2)0,
                7, 10, 11, 14, 15, 18, 19, 22, 23, 26,
                7, 10, 11, 14, (6)0 ) ;

DECLARE SS2126 ( 21:26, 6 ) FIXED BIN STATIC INIT (
                1, 2,  7, 11, 14, 19,
                0, (5)0,
                1, 3,  8, 12, 15, 19,
                1, 4,  9, 11, 14, 18,
                1, 5, 10, 13, 14, 17,
                1, 6,  8, 12, 15, 19 ) ;

DECLARE T2126 (19) CHAR (23) STATIC INIT (
                'REVISE',
                'FORM X2,4,4,10,16',
                'FORM X6,4,3,4,1',
                'FORM X2,4,9,5,2,16',
                'FORM X2,4,4,4,4,3,1,7,5',
                'FORM X6,4,9,4,1',
                'ORDER 2,1,4,5',
                'ORDER 1,53,54,55',
                'ORDER 5,1,4,3,2',
                'ORDER 2,1,3,4,5,6,7,8',
                'KEY 1',
                'KEY 1,53',
                'KEY 1,3,4',
                'INCLUDE 1',
                'INCLUDE 53',
                'INCLUDE 3',
                'FILE FACIL',
                'FILE INST',
                'FILE COURSE' ) ;

/*************************/

        ON ERROR BEGIN ;
             NONC = ONCODE ;
ER99:        NNER = 99 ;
             NUMB = NT ;
                  DISPLAY (BLANK) ;
                  DISPLAY (ERMSG) ;
                  DISPLAY (DERM) ;
                  DISPLAY (BLANK) ;
                  DISPLAY ( NUMB || '  ' || TRAN64 ) ;
             NN = 8 ;
                  GO TO SETCODE ;
             END ;

        ON CONVERSION BEGIN ;
             NONC = ONCODE ;
             NNCER = NCERR ;
ER2:         NNER = 2 ;
             NUMB = NT ;
```

Listing of Filename = SUPRPGM, Filetype = PLI (Cont'd)

```
                    DISPLAY (BLANK) ;
                    DISPLAY (ERMSG) ;
                    DISPLAY (DERM) ;
                    DISPLAY (DERM2) ;
                    DISPLAY ( NUMB || '  ' || TRAN64 ) ;
                    DISPLAY (BLANK) ;
                         GO TO CONVERTREROUTE ;
               END ;

/*************************/

STARTERRUP:
         PB = ADDR (BUF) ;
         PT = ADDR (TRAN) ;
         NT, NCERR, NER = 0 ;
         AOPEN, BOPEN, COPEN = '0'B ;

         OPEN FILE (TRANS) RECORD UPDATE ;
         ON ENDFILE (TRANS) GO TO WINDUP ;

         DISPLAY (BLANK) ;
         DISPLAY ( 'PRINT TRACE OF TRANSACTIONS ?? (Y/N)' )
              REPLY ( BUF ) ;
         IF ( BUFA(1) = 'N' ) THEN PFLAG = '0'B ;
         DISPLAY (BLANK) ;
              GO TO START ;

CONVERTREROUTE:
         IF ( NCERR = 1 ) THEN GO TO READ1A ;
         IF ( NCERR = 2 ) THEN GO TO ER1 ;

         /* NOTE - FALLTHRU IF NCERR IS NOT = 1 OR 2 */

/*************************/

START:
         READR = STEP1 ;
READ1:   NCERR = 1 ;
READ1A:  READ FILE (TRANS) INTO (TRAN) ;

         IF ( CC1 = '*' ) THEN
              DO ;
              IF ( PFLAG ) THEN DISPLAY ( F64 ) ;
                   GO TO READ1A ;
              END ;

         NT = NT + 1 ;
         TSEQ = NT ;

              REWRITE FILE (TRANS) FROM (TRAN) ;
              IF ( PFLAG ) THEN DISPLAY ( TSEQ || '     ' || F64 ) ;

         GET STRING (TNF) EDIT (NF) ( F(2) ) ;
         NCERR = 2 ;
```

Listing of Filename = SUPRPGM, Filetype = PLI (Cont'd)

```
        IF ( NF > 1 ) & ( NF < 28 )
             THEN GO TO READR ;

             GO TO ER1 ;

STEP1:       GO TO VECTOR ( NF ) ;

/*************************/

VECTOR(1):   VECTOR(4):  VECTOR(5):  VECTOR(9):
VECTOR(10):  VECTOR(22):
ER1:    /* HERE, INVALID FORMAT NUMBER IN COLUMNS 1-2 */

        NER = 1 ;
        RBUST = READ1A ;
             GO TO RUSTN ;

/*************************/

VECTOR(2):  /* LEVEL-2 CHANGE TO FACILITIES FILE (NON-KEY) */

        REOF = V2J ;  FLAGEOF = '1'B ;
        BUF = 'UPDATE' ;  SRET2 = V2A ;  GO TO WRITEC ;

V2A:    BUF = '     FILE FACIL' ;  SRET2 = V2B ;  GO TO WRITEC ;

V2B:    BUF = 'CDP=' ;
        NX = 5 ;  NA = 7 ;  NB = 10 ;
             SRET1 = V2C ;  GO TO GETF ;

V2C:    SUBSTR (BUF, NX, 5) = 'BLDG=' ;
        NX = NX + 5 ;
        NA = 11 ;  NB = 14 ;
             SRET1 = V2D ;
             WIDE = 4 ;  GO TO GETFBL ;

V2D:    SUBSTR (BUF, NX, 3) = 'RM=' ;
        NX = NX + 3 ;
        NA = 15 ;  NB = 18 ;
             SRET1 = V2E ;  GO TO GETF ;

V2E:    NA = 19 ;  NB = 24 ;
             SRET1 = V2F ;  GO TO GETF ;

V2F:    NX = NX - 1 ;
        SUBSTR (BUF, NX, 5) = '=SET=' ;
        NX = NX + 5 ;
        NA = 25 ;  NB = 31 ;
             SRET1 = V2G ;  GO TO GETF ;

V2G:    SRET3 = V2B ;  N1 = 2 ;

V2GA:   SUBSTR (BUF, NX, 1) = '3' ;
             SRET2 = V2H ;  GO TO WRITEC ;
```

Listing of Filename = SUPRPGM, Filetype = PLI (Cont'd)

```
V2H:    READR = V2I ;
            GO TO READ1 ;

V2I:    IF ( NF = N1 )
            THEN GO TO SRET3 ;
        RGEN = STEP1 ;

V2J:    BUF = 'END' ;
        FLAGEOF = '0'B ;
            SRET2 = RGEN ;
                GO TO WRITEC ;

/*************************/

VECTOR(3):  /* LEVEL-1 CHANGE, NON-KEY FIELDS, INSTRUCTOR FILE */

        REOF = V2J ;  FLAGEOF = '1'B ;
        BUF = 'UPDATE' ;  SRET2 = V3A ;  GO TO WRITEC ;

V3A:    BUF = '      FILE INST' ;
            SRET2 = V3B ;  GO TO WRITEC ;

V3B:    BUF = 'ID=' ;
        NX = 4 ;  NA = 7 ;  NB = 15 ;
            SRET1 = V3C ;  GO TO GETF ;

V3C:    NA = 16 ;  NB = 21 ;
            SRET1 = V3D ;  GO TO GETF ;

V3D:    NX = NX - 1 ;
        SUBSTR (BUF, NX, 5) = '=SET=' ;
        NX = NX + 5 ;
        NA = 22 ;  NB = 37 ;
            SRET1 = V3E ;  GO TO GETF ;

V3E:    SRET3 = V3B ;
            N1 = 3 ;
                GO TO V2GA ;

/*************************/

VECTOR(6):  /* LEVEL-1 NON-KEY FIELD CHANGE, COURSE FILE */

        COM678 = V6C ;

V6AA:   REOF = V2J ;  FLAGEOF = '1'B ;
        BUF = 'UPDATE' ;  SRET2 = V6A ;  GO TO WRITEC ;

V6A:    BUF = '      FILE COURSE' ;  SRET2 = V6B ;  GO TO WRITEC ;

V6B:    BUF = 'CDP=' ;
        NX = 5 ;  NA = 7 ;  NB = 10 ;
            SRET1 = COM678 ;  GO TO GETF ;

V6C:    NA = 11 ;  NB = 16 ;
```

Listing of Filename = SUPRPGM, Filetype = PLI (Cont'd)

```
          NXSAV = NX ;
               SRET1 = V6D ;  GO TO GETF ;

V6D:      NX = NX - 1 ;
          SUBSTR (BUF, NX, 5) = '=SET=' ;
          NX = NX + 5 ;
          NA = 17 ;  NB = 32 ;
          SRET1 = V6E ;

               NWIDE = 10 ;
          IF ( SUBSTR(BUF,NXSAV,4) = 'CTLG' )
               THEN GO TO GETFRL ;

               NWIDE = 16 ;
          IF ( SUBSTR(BUF,NXSAV,3) = 'CRS' )
               THEN GO TO GETFRL ;

               GO TO GETF ;

V6E:      SRET3 = V6B ;
               N1 = 6 ;
                    GO TO V2GA ;

/*************************/

VECTOR(7):  /* COURSE FILE, SIR-SEGMENT, NON-KEY FIELD CHANGE */

          COM678 = V7C ;
               GO TO V6AA ;

V7C:      SUBSTR (BUF, NX, 3) = 'F1=' ;
          NX = NX + 3 ;
          NA = 11 ;  NB = 13 ;
               SRET1 = V7D ;  GO TO GETF ;

V7D:      NA = 14 ;  NB = 19 ;
               SRET1 = V7E ;  GO TO GETF ;

V7E:      NX = NX - 1 ;
          SUBSTR (BUF, NX, 5) = '=SET=' ;
          NX = NX + 5 ;
          NA = 20 ;  NB = 23 ;
               SRET1 = V7F ;  GO TO GETF ;

V7F:      SRET3 = V6B ;
               N1 = 7 ;
                    GO TO V2GA ;

/*************************/

VECTOR(8):  /* ID-SEGMENT NON-KEY FIELD CHANGE, COURSE FILE */

          COM678 = V8C ;
               GO TO V6AA ;
```

Listing of Filename = SUPRPGM, Filetype = PLI (Cont'd)

```
V8C:    SUBSTR (BUF, NX, 3) = 'F1=' ;
        NX = NX + 3 ;
        NA = 11 ;  NB = 19 ;
             SRET1 = V8D ;  GO TO GETF ;

V8D:    NA = 20 ;  NB = 25 ;
             SRET1 = V8E ;  GO TO GETF ;

V8E:    NX = NX - 1 ;
        SUBSTR (BUF, NX, 5) = '=SET=' ;
        NX = NX + 5 ;
        NA = 26 ;  NB = 29 ;
             SRET1 = V8F ;  GO TO GETF ;

V8F:    SRET3 = V6H ;
             N1 = 8 ;
                  GO TO V2GA ;

/*************************/

GETF:   /* G.P FIELD ISOLATION S.R.; RETURN LINK IS (SRET1).
           ENTER WITH NA, NB (TRAN REC) AND NX (BUF REC) SET.
           LEAVE WITH NX SET TO NEXT AVAIL SLOT IN (BUF).
           FILL EITHER COMMA OR BLANK, (FILLF) DETERMINES. */

        FILLF = ',' ;
             GO TO GLP1 ;

GETFA:  FILLF = ' ' ;

GLP1:   NX = NX - 1 ;
GLP2:   IF ( TRANSA(NA) = ' ' ) THEN GO TO GFTALLY ;

             NX = NX + 1 ;
             BUFA (NX) = TRANSA (NA) ;

GFTALLY: NA = NA + 1 ;
         IF ( NA <= NB ) THEN GO TO GLP2 ;

PUTCHAR:     NX = NX + 1 ;
             BUFA (NX) = FILLF ;
             NX = NX + 1 ;

                  GO TO SRET1 ;

/*************************/

GETFRL: /* G.P FIELD ISOLATION S.R.; RETURN LINK IS (SRET1).
           ENTER WITH NA, NB (TRAN REC) AND NX (BUF REC) SET.
           LEAVE WITH NX SET TO NEXT AVAIL SLOT IN (BUF).
           FILL EITHER COMMA OR BLANK, (FILLF) DETERMINES.
           ENTRY MADE WITH NWIDE = MAX ALLOWED FIELD WIDTH OF
           CHARACTER FIELD. */

        FILLF = ',' ;
```

Listing of Filename = SUPRPGM, Filetype = PLI (Cont'd)

```
                GO TO GFBL1 ;

GETFBLA:        FILLF = ' ' ;

GFBL1:  NFILL = 0 ;
        NX = NX - 1 ;

GFBL2:  NX = NX + 1 ;
        NFILL = NFILL + 1 ;
        BUFA(NX) = TRANSA(NA) ;

        NA = NA + 1 ;
        IF ( NFILL >= NWIDE ) THEN GO TO PUTCHAR ;
        IF ( NA <= NB ) THEN GO TO GFBL2 ;

                GO TO PUTCHAR ;     /* IN (GETF) S.R.. ABOVE */

/*************************/

VECTOR(11):  /* DELETE AN OLD CDP TRANSACTION */

        COM678 = V11C ;
V11A:   REOF = WIND2 ;  FLAGEOF = '0'B ;
        BUF = 'DELETE' ;  SRET2 = V11B ;  GO TO WRITEA ;

V11B:   BUF = '      FILE FACIL' ;  SRET2 = COM678 ;  GO TO WRITEA ;

V11C:   COM678 = V11E ;
V11D:   BUF = 'CDP=OUT=' ;
        NX = 9 ;  NA = 7 ;  NB = 10 ;
                SRET1 = COM678 ;  GO TO GETF ;

V11E:   SRET3 = V11H ;
V11F:   SUBSTR (BUF, NX, 1) = '$' ;
                SRET2 = V11G ;  GO TO WRITEA ;

V11G:   BUF = 'END' ;
                SRET2 = SRET3 ;  GO TO WRITEA ;

V11H:   COM678 = V11L ;
V11J:   REOF = WIND2 ;  FLAGEOF = '0'B ;
        BUF = 'DELETE' ;  SRET2 = V11K ;  GO TO WRITEA ;

V11K:   BUF = '      FILE COURSE' ;  SRET2 = COM678 ;  GO TO WRITEA ;

V11L:   COM678 = V11M ;
                GO TO V11D ;

V11M:   SRET3 = START ;
                GO TO V11F ;

/*************************/

VECTOR(12):  /* DELETE AN OLD SIR-SEGMENT TRANSACTION */
```

Listing of Filename = SUPRPGM, Filetype = PLI (Cont'd)

```
          COM678 = V12A ;
               GO TO V11J ;

V12A:     COM678 = V12C ;
V12B:     BUF = 'CDP=' ;
          NX = 5 ;  NA = 7 ;  NB = 10 ;
               SRET1 = COM678 ;  GO TO GETF ;

V12C:     SUBSTR (BUF, NX, 7) = 'F1=OUT=' ;
          NX = NX + 7 ;
          NA = 11 ;  NB = 13 ;

               SRET1 = V11M ;
                    GO TO GETF ;

/*************************/

VECTOR(14):  /* DELETE AN EXISTING BLDG/RM FROM A CDP */

          COM678 = V14A ;
               GO TO V11A ;

V14A:     COM678 = V14B ;
               GO TO V12B ;

V14B:     SUBSTR (BUF, NX, 3) = 'RM=' ;
          NX = NX + 3 ;
          NA = 15 ;  NB = 18 ;
               SRET1 = V14C ;  GO TO GETF ;

V14C:     SUBSTR (BUF, NX, 9) = 'BLDG=OUT=' ;
          NX = NX + 9 ;
          NA = 11 ;  NB = 14 ;

V14D:          SRET1 = V11M ;
               NWIDE = 4 ;
                    GO TO GETFBL ;

/*************************/

VECTOR(27):  /* DELETE AN EXISTING ID FROM A CDP */

          COM678 = V27A ;
               GO TO V11J ;

V27A:     COM678 = V27B ;
               GO TO V12B ;

V27B:     SUBSTR (BUF, NX, 7) = 'F1=OUT=' ;
          NX = NX + 7 ;
          NA = 11 ;  NB = 19 ;

               SRET1 = V11M ;
                    GO TO GETF ;
```

AD-A040 238 IBM CORP CAPE CANAVERAL FLA F/G 5/9
DESIGN OF TRAINING SYSTEMS, TRAPAC USER'S MANUAL, DATA BASE, SC--ETC(U)
SEP 76 K V BRANCH, J D STALEY, R E YANKO N61339-73-C-0097
UNCLASSIFIED TAEG-36 NL

5 OF 6
AD A040238



Listing of Filename = SUPRPGM, Filetype = PLI (Cont'd)

```
/*************************/

VECTOR(13):  /* DELETE AN EXISTING INSTRUCTOR ID */
VECTOR(15):  /* DELETE AN ENTIRE SCHOOL */
VECTOR(16):  /* CHANGE AN EXISTING CDP NUMBER */
VECTOR(17):  /* CHANGE AN EXISTING SIR */
VECTOR(18):  /* CHANGE AN EXISTING INSTRUCTOR ID NUMBER */
VECTOR(19):  /* CHANGE AN EXISTING BLDG/RM INDENTIFIER */
VECTOR(20):  /* CHANGE AN EXISTING SCHOOL NUMBER */

        NTRAN = NT ;
        NUM2 = NF ;
        DS1320 = 'M' || NUM2 || 'EXA' ;

        IF ( DS1320 ¬= DSOPEN ) THEN
L1320A:     DO ;
            IF ( DSOPEN ¬= '          ' )
                THEN CLOSE FILE (DSGEN) ;
            OPEN FILE (DSGEN)
                TITLE (DS1320) RECORD OUTPUT ;
            DSOPEN = DS1320 ;
            END L1320A ;

        BUF = '&TYPE OFF' ;
            SRET2 = V1320A ;  GO TO WDSGEN ;

V1320A: BUF = 'EXEC M' || NUM2 || 'EXB ' || NTRAN ;

            NX = 17 ;
            J = 1 ;

LOOP1320:   NA = SS1320 ( NF, J ) ;
            IF ( NA = 0 ) THEN GO TO V1320B ;
            NB = SS1320 ( NF, J+1 ) ;
                NX = NX + 1 ;
                SRET1 = TALLY1320 ;
                    GO TO GETFA ;

TALLY1320:  J = J + 2 ;
            IF ( J < JMAX ) THEN GO TO LOOP1320 ;

V1320B: SRET2 = START ;
            GO TO WDSGEN ;


/*************************/

VECTOR(21):  /* ADD A NEW CDP */
VECTOR(23):  /* ADD A NEW SIR-SEGMENT */
VECTOR(24):  /* ADD A NEW INSTRUCTOR TO THE INSTRUCTOR FILE */
VECTOR(25):  /* ADD A NEW BLDG/RM TO AN EXISTING CDP */
VECTOR(26):  /* ADD A NEW ID-SEGMENT TO AN EXISTING CDP */

        N1 = NF ;
        REOF = V2126C ;
```

Listing of Filename = SUPRPGM, Filetype = PLI (Cont'd)

```
          FLAGEOF = '1'B ;

          IF ( ¬BOPEN ) THEN
               DO ;
               OPEN FILE (DSB) RECORD OUTPUT ;
                    BOPEN = '1'B ;
               END ;

               J = 1 ;
LOOP2126:      BUF = T2126 ( SS2126 ( NF, J ) ) ;

                    SRET2 = V2126AA ;  GO TO WRITEB ;

V2126AA:       J = J + 1 ;
               IF ( J <= 6 ) THEN GO TO LOOP2126 ;

V2126A: BUF = TRAN ;
               SRET2 = V2126D ;  GO TO WRITEB ;

V2126D: READR = V2126H ;
               GO TO READ1 ;

V2126B: IF ( NF = N1 ) THEN GO TO V2126A ;

          RGEN = STEP1 ;
V2126C:   BUF = 'END' ;
               SRET2 = RGEN ;

                    GO TO WRITEH ;

/*************************/

               /* DATASET HANDLER SUBROUTINES IN THIS SECTION:
                  COMMON RETURN VECTOR IS (SRET2) */

WRITEC: /* TRANSACTION TYPES 2, 3, 6, 7, 8 (CHANGES) */
          IF (COPEN ) THEN
               DO ;
               OPEN FILE (DSC) RECORD OUTPUT ;
               COPEN = '1'B ;
               END ;

                    WRITE FILE (DSC) FROM (BUF) ;

WRRET:    IF ( PFLAG ) THEN DISPLAY ( '          ' || F40 ) ;

               GO TO SRET2 ;

WRITEA: /* TRANSACTION TYPES 11, 12, 14, 27 (DELETES) */
          IF (AOPEN ) THEN
               DO ;
               OPEN FILE (DSA) RECORD OUTPUT ;
               AOPEN = '1'B ;
               END ;
```

Listing of Filename = SUPRPGM, Filetype = PLI (Cont'd)

```
                    WRITE FILE (DSA) FROM (BUF) ;

              GO TO WRRET ;

WRITEB: /* TRANSACTION TYPES 21, 23, 24, 25, 26 */

              WRITE FILE (DSB) FROM (BUF) ;

                   GO TO WRRET ;

WDSGEN: /* TRANSACTION TYPES 13, 15, 16, 17, 18, 19, 20X */

              WRITE FILE (DSGEN) FROM (BUF) ;

                   GO TO WRRET ;

/*************************/

BUSTN:  /* ERROR MESSAGE OUTPUT S.R.;
           RETURN LINK IS (RBUST);
           ERROR NUMBER IS IN (NER) UPON ENTRY */

        DISPLAY (BLANK) ;
        NNER = NER ;
        DISPLAY (ERMSG) ;
        NUMB = NT ;
        DISPLAY ( NUMB || ' ' || TRAN64 ) ;
              GO TO RBUST ;

/*************************/

WINDUP: /* EOF ON TRANSACTION INPUT FILE; HERE */
        NN = 0 ;

        IF ( FLAGEOF ) THEN
WLP:         DO ;
             RGEN = WIND2 ;
                  GO TO REOF ;
             END WLP ;

WIND2:  CLOSE FILE (DSA), FILE (DSB), FILE (DSC) ;

        IF ( DSOPEN= '          ' )
             THEN CLOSE FILE (DSGEN) ;

SETCODE:  CALL IHESARC ( NN ) ;

              END SUPRPGM ;
```

GENERAL PURPOSE DATA BASE EXECUTIVES

Three general purpose executive sequences have been developed to assist the user with common data base maintenance tasks. These executives bear the names of the files with which they are associated, i.e., FACIL, INST, and COURSE. As with all NCSS executives, the sequence is initiated by typing in the executive name. These three executives require one (and only one) execution-time arguments. The following indicates the selections:

Executive Name	Arguments
FACIL	DUMP
INST	RELOAD
COURSE	FNS
	INDEX
	DESC
	DESCLOAD

As stated above, the execution arguments are mutually exclusive and must be invoked one at a time. Selection of the executive by the same name as the RAMIS file concerned is self-explanatory. The arguments elicit the following actions:

DUMP - A RAMIS dump function is executed against the selected file. In the case of the COURSE and INST files, auxiliary storage is used due to the size of the files. When completed, the file dump will be reformatted (not required for the FACIL file) and the resulting records will be stored in on-line disk storage. A subsequent invocation of the same executive with the RELOAD option (explained below) uses this reformatted dump file to erase the file concerned and effect its complete reload. The reformatted dump records reside in the following files (filename/filetype):

Facilities File - FACIL/DUMP

Instructor File - INST/DUMP2

Course File - COURSE/DUMP2

RELOAD - Using the dump file created by the DUMP option as explained above, this argument forces the erase and complete reload of the file concerned. After successful execution, the RAMIS file will be in the exact state as existed at the time that the DUMP was made.

FNS - This option causes an on-line listing of the executive sequences associated with these three general purpose executives; e.g., the following entry:

course fns CR

will cause the listing of executives E7, E8, and E9. The FNS option is really only of use to those concerned with system or program maintenance.

INDEX - This argument causes a RAMIS listing of the selected file's internal RAMIS description.

DESC - This option lists the selected file's RAMIS description as it exists in card-image form. For each of the three files, the applicable description is stored in the NCSS system as follows (filename/filetype):

Facilities File - FACIL/DESC

Instructor File - INST/DESC

Course File - COURSE/DESC

DESCLOAD - This entry causes the applicable file's description to be entered into the RAMIS system. This action completely erases the previous file description as well as the data file itself. This erase action is complete and the previous description and file contents will be irretrievably lost. If the file description (found in filename/DESC) has been modified before DESCLOAD, then a subsequent attempt at RELOAD will fail unless the related RAMIS functions associated with RELOAD have been modified in conjunction with the description change.

The following three executives utilize the reformatted dump files FACIL/DUMP, INST/DUMP2, and COURSE/DUMP2 to produce a listing of the current file contents. All fields in all records are listed. No arguments are required in the operation of these executives. At the time of this document, all printer listings were being routed to the San Diego office of National CSS. The user may pick up the printer output at this location.

DOTS File	Printer DUMP Executive Name
Facilities	PFACIL
Instructor	PINST
Course	PCOURSE

To assist the user in effective data base maintenance, a set of forms can be produced via printer. These forms can be used in a fill-in-the-blanks manner

to note the requirement for data base changes. No arguments are required in the invocation of these executives. No forms are provided for the Facilities file.

DOTS File	Blank Forms Executive Name
Instructor	FINST
Course	FCOURSE

Listings of the executive sequences, RAMIS functions, and RAMIS run procedures that are associated with the establishment and modification of the three DOTS files are on the pages following. Several of these software components are used in more than one sequence. Duplicate listings are considered superfluous in this instance and, in general, are not provided.

Listing of Filename = COURSE, Filetype = EXEC

```
      &TYPE OFF
      &SPACE
      &IF &INDEX GT 0 &GOTO -S1
         &PRINT ?? NO FUNCTION, NO ACTION ??
         &QUIT
 -S1 EXEC GP1 &1 COURSE F3 E8 E9 E7
      &QUIT
```

Listing of Filename = E1, Filetype = EXEC

```
      &TYPE OFF
      &COMMENT RELOAD PROCEDURE FOR FACIL
      &COMMENT THIS IS NEW/NEW FORMAT LOAD.
      &COMMENT USE FACIL1 / RAM FOR OLD/NEW LOAD
 EXEC RAM FACIL2 RAM
      &EXIT
```

Listing of Filename = E2, Filetype = EXEC

```
     &TYPE OFF
     &COMMENT LIST FUNCTIONS OF FACIL FILE
LISTF FACIL* * (A)
EXEC GPLISTF E1 E2 E3
     &QUIT
```

Listing of Filename = E3, Filetype = EXEC

```
     &TYPE OFF
     &COMMENT DUMP / REFORMAT FOR FACIL.
     &COMMENT ARGS ARE : (1) LAST FIELD NAME FOR DUMP //
EXEC NDUMP FACIL &1 FACIL DUMP P
     &EXIT
```

Listing of Filename = E4, Filetype = EXEC

```
     &TYPE OFF
     &COMMENT DUMP AND REFORMAT FOR INST
     &COMMENT THIS IS FOR NEW INST FORMAT
EXEC NDUMP INST &1 INST DUMP U TEMP5
EXEC REFORM INST 160 800 2 U
     &EXIT
```

Listing of Filename = E5, Filetype = EXEC

```
     &TYPE OFF
     &COMMENT RELOAD FOR INST FROM INST / DUMP2
     &COMMENT THIS IS NEW / NEW FORMAT. SEE
     &COMMENT INST1 / RAM FOR OLD / NEW FORMAT RELOAD.
EXEC RAM INST2 RAM
     &EXIT
```

Listing of Filename = E6, Filetype = EXEC

```
        &TYPE OFF
        &COMMENT THIS IS FNS FUNCTION FOR INST
    LISTF INST* * * (A)
    EXEC GPLISTF E4 E5 E6
        &QUIT
```

Listing of Filename = E7, Filetype = EXEC

```
        &TYPE OFF
        &COMMENT DUMP / REFORMAT FOR NEW COURSE FILE
    EXEC NDUMP COURSE A1 COURSE DUMP U TEMP5
    EXEC REFORM COURSE 240 720 3 U
        &EXIT
```

Listing of Filename = E8, Filetype = EXEC

```
        &TYPE OFF
        &COMMENT THIS IS RELOAD FOR COURSE FILE.
        &COMMENT NOTE - NEW DUMP FORMAT / NEW COURSE FORMAT.
    EXEC PAM COURSE3 RAM
        &EXIT
```

Listing of Filename = E9, Filetype = EXEC

```
        &TYPE OFF
        &COMMENT THIS IS FNS FUNCTION FOR COURSE.
    LISTF COURSE* * * (A)
    EXEC GPLISTF E7 E8 E9
        &QUIT
```

Listing of Filename = FACIL, Filetype = EXEC

```
      &TYPE OFF
      &SPACE
      &IF &INDEX GT 0 &GOTO -S1
         &PRINT ?? NO FUNCTION, NO ACTION ??
         &QUIT
-S1 EXEC GP1 &1 FACIL AVHRS E1 E2 E3
      &QUIT
```

Listing of Filename = FCOURSE, Filetype = EXEC

```
      &TYPE OFF
FILEDEF SYSPRINT PTR
NETROUTE E SU
SET TADDR ADDRESS MEMO2
LOAD FCOURSE (CLEAR LIBE) PLILIB
START (BRIEF)
FILEDEF SYSPRINT CLEAR
NETROUTE E ATL
SET TADDR ADDRESS MEMO
      &EXIT
```

Listing of Filename = FINST, Filetype = EXEC

```
      &TYPE OFF
FILEDEF SYSPRINT PTR
NETROUTE E SU
SET TADDR ADDRESS MEMO2
LOAD FINST (CLEAR LIBE) PLILIB
START (BRIEF)
FILEDEF SYSPRINT CLEAR
NETROUTE E ATL
SET TADDR ADDRESS MEMO
      &EXIT
```

Listing of Filename = GPLISTF, Filetype = EXEC

```
     &TYPE OFF
     &COMMENT G.P. 1 TO 5 FILE LISTF
     &IF &INDEX LT 1 &EXIT
LISTF &1 EXEC (A)
     &IF &INDEX LT 2 &EXIT
LISTF &2 EXEC (A)
     &IF &INDEX LT 3 &EXIT
LISTF &3 EXEC (A)
     &IF &INDEX LT 4 &EXIT
LISTF &4 EXEC (A)
     &IF &INDEX LT 5 &EXIT
LISTF &5 EXEC (A)
     &EXIT
```

Listing of Filename = GP1, Filetype = EXEC

```
     &TYPE OFF
     &COMMENT PARMS ARE : (1) OPS KEYWORD / (2) RAMIS FN /
     &COMMENT (3) LAST FIELDNAME FOR RAMIS DUMP /
     &COMMENT (4) EXEC NAME FOR RELOAD /
     &COMMENT (5) EXEC NAME FOR FNS LIST /
     &COMMENT (6) DUMP AND REFORMAT EXEC NAME //
     &IF &INDEX NE 6 &QUIT
-S1  &IF &1 NE MENU &GOTO -S2
        PRINTF &2 MENU
        &EXIT
-S2  &IF &1 NE DUMP &GOTO -S3
        EXEC &6 &3
        &EXIT
-S3  &IF &1 NE RELOAD &GOTO -S4
        EXEC &4
        &EXIT
-S4  &IF &1 NE FILEDEF &GOTO -S5
        PRINTF &2 FILEDEF
        &EXIT
-S5  &IF &1 NE FNS &GOTO -S6
        EXEC &5
        &EXIT
-S6  &IF &1 NE INDEX &GOTO -S7
        EXEC RUNPROC INDEX &2
        &EXIT
-S7  &IF &1 NE DESC &GOTO -S8
        PRINTF &2 DESC
        &EXIT
-S8  &IF &1 NE DESCLOAD &GOTO -S9
        EXEC RAM &2 DESC
        &EXIT
-S9  &SPACE
     &PRINT ??? COMMAND FUNCTION &1 NOT UNDERSTOOD ???
     &PRINT      TRY AGAIN...
     &SPACE
     &QUIT
```

Listing of Filename = INST, Filetype = EXEC

```
      &TYPE OFF
      &SPACE
      &IF &INDEX GT 0 &GOTO -S1
         &PRINT ?? NO FUNCTION, NO ACTION ??
         &QUIT
 -S1 EXEC GP1 &1 INST SLIB E5 E6 E4
      &QUIT
```

Listing of Filename = NDUMP, Filetype = EXEC

```
      &TYPE OFF
      &COMMENT IN THIS EXEC, ALL ARGS ARE PROVIDED.
      &COMMENT ALL RAMIS MSGS ARE SUPPRESSED.
      &COMMENT ARGS ARE : RAMIS FILE NAME / LAST FIELD TO DUMP /
      &COMMENT DUMP FN / DUMP FT / DUMP FM /
      &COMMENT TEMPN FOR SPACE ALLOCATE //
      &IF &INDEX LT 5 &QUIT
     &IF &5 EQ P &GOTO -S1
     &IF &5 NE D &QUIT
ATTACH &6 AS 193 BRIEF
-S1 FILEDEF RAMUTIL1 DSK &3 &4 &5 RE FB LR 80 BL 800
     &SPACE
     &PRINT &1 FILE DUMP IS STARTING
     &SPACE
      &STACK KT
      &STACK DUMP
      &STACK FILE &1
      &STACK PRINT &2 AND SAVE
      &STACK END
      &STACK FIN
ATTACH RAMIS ALL BR
FILEDEF 6 MSG KILL
RAMIS
DETACH RAMIS BR
     &STACK KT
FILEDEF RAMUTIL1 CLEAR
FILEDEF 6 CONO
     &STACK KT
     &STACK L /D=/
     &STACK Q
EDIT MSG KILL
     &SPACE
     &PRINT &1 FILE DUMPED TO &5 DISK
     &SPACE
LISTF &3 &4 &5 (A)
     &EXIT
```

Listing of Filename = PCB, Filetype = EXEC

```
     &TYPE OFF
ATTACH TEMP20 AS 193 BRIEF
     &STACK 5 8 1 4 227 235
SORT COURSE DUMP2 COURSE PRINT (BRIEF WMODE=U)
LISTF COURSE DUMP2 P (A)
LISTF COURSE PRINT U (A)
FILEDEF DUMP DSK COURSE PRINT U RE F LR 240 BL 240
FILEDEF SYSPRINT PTR
LOAD PCOURSE (CLEAR LIBE) PLILIB
START (BRIEF)
FILEDEF SYSPRINT CLEAR
FILEDEF DUMP CLEAR
DETACH 193 BRIEF
```

Listing of Filename = PCOURSE, Filetype = EXEC

```
     &TYPE OFF
ATTACH TEMP20 AS 193 BRIEF
     &STACK 5 8 1 4 227 235
SORT COURSE DUMP2 COURSE PRINT (BRIEF WMODE=U)
LISTF COURSE DUMP2 P (A)
LISTF COURSE PRINT U (A)
NETROUTE E SD
SET TADDR ADDRESS MEMO2
FILEDEF DUMP DSK COURSE PRINT U RE F LR 240 BL 240
FILEDEF SYSPRINT PTR
LOAD PCOURSE (CLEAR LIBE) PLILIB
START (BRIEF)
FILEDEF SYSPRINT CLEAR
FILEDEF DUMP CLEAR
DETACH 193 BRIEF
NETROUTE E ATL
SET TADDR ADDRESS MEMO
     &EXIT
```

Listing of Filename = PFACIL, Filetype = EXEC

```
     &TYPE OFF
ATTACH TEMP10 AS 193 BRIEF
     &STACK 5 8 1 4
SORT FACIL DUMP FACIL PRINT (BRIEF WMODE=U)
LISTF FACIL DUMP P (A)
LISTF FACIL PRINT U (A)
NETROUTE E SD
SET TADDR ADDRESS MEMO2
FILEDEF DUMP DSK FACIL PRINT U RE F LR 80 BL 80
FILEDEF SYSPRINT PTR
LOAD PFACIL (CLEAR LIBE) PLILIB
START (BRIEF)
FILEDEF SYSPRINT CLEAR
FILEDEF DUMP CLEAR
DETACH 193 BRIEF
NETROUTE E ATL
SET TADDR ADDRESS MEMO
&EXIT
```

Listing of Filename = PFCB, Filetype = EXEC

```
    &TYPE OFF
ATTACH TEMP10 AS 193 BRIEF
    &STACK 5 8 1 4
SORT FACIL DUMP FACIL PRINT (BRIEF WMODE=U)
LISTF FACIL DUMP P (A)
LISTF FACIL PRINT U (A)
FILEDEF DUMP DSK FACIL PRINT U RE F LR 80 BL 80
FILEDEF SYSPRINT PTR
LOAD PFACIL (CLEAR LIBE) PLILIB
START (BRIEF)
FILEDEF SYSPRINT CLEAR
FILEDEF DUMP CLEAR
DETACH 193 BRIEF
```

Listing of Filename = PICB, Filetype = EXEC

```
    &TYPE OFF
ATTACH TEMP20 AS 193 BRIEF
    &STACK 33 36 1 9
SORT INST DUMP2 INST PRINT (BRIEF WMODE=U)
LISTF INST DUMP2 P (A)
LISTF INST PRINT U (A)
FILEDEF DUMP DSK INST PRINT U RE F LR 160 BL 160
FILEDEF SYSPRINT PTR
LOAD PINST (CLEAR LIBE) PLILIB
START (BRIEF)
FILEDEF SYSPRINT CLEAR
FILEDEF DUMP CLEAR
DETACH 193 BRIEF
```

Listing of Filename = PINST, Filetype = EXEC

```
    &TYPE OFF
ATTACH TEMP20 AS 193 BRIEF
    &STACK 33 36 1 9
SORT INST DUMP2 INST PRINT (BRIEF WMODE=U)
LISTF INST DUMP2 P (A)
LISTF INST PRINT U (A)
NETROUTE E SU
SET TADDR ADDRESS MEMO2
FILEDEF DUMP DSK INST PRINT U RE F LR 160 BL 160
FILEDEF SYSPRINT PTR
LOAD PINST (CLEAR LIBE) PLILIB
START (BRIEF)
FILEDEF SYSPRINT CLEAR
FILEDEF DUMP CLEAR
DETACH 193 BRIEF
NETROUTE E ATL
SET TADDR ADDRESS MEMO
    &EXIT
```

Listing of Filename = PROCESS, Filetype = EXEC

```
      &TYPE OFF
      &SPACE
      STATE &1 &2 P
      &IF &INDEX0 EQ 0 &GOTO -S1
      &PRINT SKIPPING DATASET &1 / &2 PROCESSING
      &EXIT
      -S1 &PRINT NOW PROCESSING DATASET &1 / &2
      &SPACE
      PRINTF &1 &2 P
      &SPACE
      &IF &2 EQ RAM &GOTO -S2
      EXEC &1
      &EXIT
      -S2 EXEC RAM &1 &2
      &EXIT
```

Listing of Filename = RAM, Filetype = EXEC

```
      &TYPE OFF
      &COMMENT PARMS ARE FILENAME / FILETYPE
      &IF &INDEX NE 2 &QUIT
      &STACK KT
 ATTACH RAMIS ALL BR
      &STACK PT
 EXEC RAMISIN &1 &2
 DETACH RAMIS BR
      &EXIT
```

Listing of Filetype = RDUMP, Filetype = EXEC

```
      &TYPE OFF
      &COMMENT IN THIS EXEC, ALL ARGS ARE PROVIDED
     &COMMENT RAMIS FILE NAME / LAST FIELD TO DUMP /
     &COMMENT DUMP FN / DUMP FT / DUMP FM
      &IF &INDEX NE 5 &QUIT
-S1 FILEDEF RAMUTIL1 DSK &3 &4 &5 RE FB LR 80 BL 800
      &STACK KT
      &STACK DUMP
      &STACK FILE &1
      &STACK PRINT &2 AND SAVE
      &STACK END
      &STACK FIN
ATTACH RAMIS ALL BR
      &STACK KT
RAMIS
DETACH RAMIS BR
FILEDEF RAMUTIL1 CLEAR
LISTF &3 &4 &5 (A)
      &EXIT
```

Listing of Filename = REFORM, Filetype = EXEC

```
     &TYPE OFF
     &COMMENT G.P. REFORMAT FOR DUMP FILES: COMBINES
     &COMMENT 2/MORE CARD IMAGES INTO SINGLE RECORD.
     &COMMENT PARMS : (1) RAMIS FN / (2) REFORM LRECL /
     &COMMENT (3) REFORM BLKSIZE / (4) REFORM BLOCKING FACTOR /
     &COMMENT (5) DUMP FILE MODE //
     &SPACE
     &IF &INDEX NE 5 &QUIT
     &PRINT STARTING REFORMAT OF DUMP FILE FOR &1
FILEDEF * CLEAR
STATE &1 DUMP &5
     &IF &INDEX0 EQ 0 &GOTO -S1
        &SPACE
        &PRINT !!! &1 / DUMP FILE DOES NOT EXIST !!!
-SER        &PRINT      FUNCTION IS TERMINATING ABNORMALLY
-SOUT       &SPACE
FILEDEF * CLEAR
        &EXIT
-S1 STATE &1 DUMP2
     &IF &INDEX0 NE 0 &GOTO -S2
ERASE &1 DUMP2
-S2 FILEDEF DUMP DSK &1 DUMP &5 RE FB LR 80 BL 800
FILEDEF DUMP2 DSK &1 DUMP2 P RE FB LR &2 BL &3
LOAD MPGM3 (CLEAR LIBE) PLILIB
START IHECMS &4
     &SPACE
     &IF &INDEX0 EQ 0 &GOTO -S3
        &PRINT !!! REFORMAT OF &1 DUMP FAILED !!!
        &PRINT      RETURN INDEX0 WAS &INDEX0
        &GOTO -SER
-S3 &PRINT REFORMAT FOR &1 COMPLETED SUCCESSFULLY
     &PRINT NORMAL RELOAD OPERATIONS MAY PROCEED
     &SPACE
LISTF &1 DUMP2 P (A)
ERASE &1 DUMP &5
     &IF &5 EQ P &GOTO -SOUT
DETACH &5 BRIEF
     &GOTO -SOUT
```

Listing of Filename = RELEASE, Filetype = EXEC

```
     &TYPE OFF
     FILEDEF DSA CLEAR
     FILEDEF DSB CLEAR
     FILEDEF DSC CLEAR
     &INDEX1 = 13
     -LOOP &NN = M !! &INDEX1
     &NN = &NN !! EXA
     FILEDEF &NN CLEAR
     &INDEX1 + 1
     &IF &INDEX1 LE 20 &GOTO -LOOP
     &EXIT
```

Listing of Filename = RUNPROC, Filetype = EXEC

```
     &TYPE OFF
     &COMMENT PARMS ARE RUNPROC NAME / 5 OPT PARMS...
     &IF &INDEX LT 1 &QUIT
     &IF &INDEX GT 6 &QUIT
     &STACK KT
     &STACK EXEC &1 &2 &3 &4 &5 &6
     &STACK FIN
ATTACH RAMIS ALL BP
EXEC RAMIS
DETACH RAMIS BR
     &STACK RT
     &EXIT
```

Listing of Filename = SETUP, Filetype = EXEC

```
     &TYPE OFF
     STATE DSA RAM P
     &IF &INDEX0 NE 0 &GOTO -S1
     ERASE DSA RAM P
     -S1 FILEDEF DSA DSK DSA RAM LR 80 RE F BL 80
     STATE DSB RAM P
     &IF &INDEX0 NE 0 &GOTO -S2
     ERASE DSB RAM P
     -S2 FILEDEF DSB DSK DSB RAM LR 80 RE F BL 80
     STATE DSC RAM P
     &IF &INDEX0 NE 0 &GOTO -S3
     ERASE DSC RAM P
     -S3 FILEDEF DSC DSK DSC RAM LR 80 RE F BL 80
     &INDEX1 = 13
     -LOOP &NN = M !! &INDEX1
     &NN = &NN !! EXA
     STATE &NN EXEC P
     &IF &INDEX0 NE 0 &GOTO -LS1
     ERASE &NN EXEC P
     -LS1 FILEDEF &NN DSK &NN EXEC LR 80 RE F BL 80
     &INDEX1 + 1
     &IF &INDEX1 LE 20 &GOTO -LOOP
     &EXIT
```

Listing of Filename = SUPER, Filetype = EXEC

```
&TYPE OFF
EXEC SETUP
&IF &INDEX NE 0 &GOTO -S2
&ARGS TYPE2 TRANS
-S2 FILEDEF TRANS DSK &1 &2 LR 80 BL 80 RE F
LOAD SUPRPGM (CLEAR LIBE) PLILIB
START (BRIEF)
&PARM = &INDEX0
FILEDEF TRANS CLEAR
EXEC RELEASE
&SPACE
&IF &PARM EQ 0 &GOTO -S1
&PRINT TRANSACTION PROCESSOR COMPLETION WAS ABNORMAL...
&PRINT COMPLETION CODE WAS &PARM
&QUIT
-S1 &PRINT RAMIS PROCESSING STARTS AT THIS POINT
EXEC PROCESS DSA RAM
EXEC PROCESS M13EXA EXEC
EXEC PROCESS M15EXA EXEC
EXEC PROCESS DSB RAM
EXEC PROCESS M16EXA EXEC
EXEC PROCESS M17EXA EXEC
EXEC PROCESS M18EXA EXEC
EXEC PROCESS M19EXA EXEC
EXEC PROCESS M20EXA EXEC
EXEC PROCESS DSC RAM
&SPACE
&PRINT RAMIS PROCESSING HAS CONCLUDED
&QUIT
```

Listing of Filename = SUPERDUM, Filetype = EXEC

```
&TYPE OFF
&COMMENT IN THIS EXEC, ALL ARGS ARE PROMPTED
&SPACE
&PRINT GIMMIE FILENAME / LAST FIELD NAME
&PRINT        DSK FILENAME / FILETYPE / FILEMODE
&SPACE
&READ ARGS
&SPACE
&IF &INDEX NE 5 &QUIT
&IF &1 EQ QUIT &QUIT
-S1 FILEDEF RAMUTIL1 DSK &3 &4 &5 RE FB LR 80 BL 800
&STACK KT
&STACK DUMP
&STACK FILE &1
&STACK PRINT &2 AND SAVE
&STACK END
&STACK FIN
ATTACH RAMIS ALL BR
&STACK KT
RAMIS
DETACH RAMIS BR
FILEDEF RAMUTIL1 CLEAR
LISTF &3 &4 &5 (A)
&EXIT
```

Listing of Filename = SUPERX, Filetype = EXEC

```
      &TYPE OFF
      EXEC SETUP
      &IF &INDEX NE 0 &GOTO -S2
      &ARGS TYPE2 TRANS
      -S2 FILEDEF TRANS DSK &1 &2 LR 80 BL 80 RF F
      LOAD SUPRPGM (CLEAR LIBE) PLILIB
      START (BRIEF)
      &PARM = &INDEX0
      FILEDEF TRANS CLEAR
      EXEC RELEASE
      &SPACE
      &IF &PARM EQ 0 &GOTO -S1
      &PRINT TRANSACTION PROCESSOR COMPLETION WAS ABNORMAL...
      &PRINT COMPLETION CODE WAS &PARM
      -S1 &QUIT
```

Listing of Filename = SUPRDUMP, Filetype = EXEC

```
      &TYPE ALL
      &COMMENT IF ARGS ARE NOT PROVIDED, THEY ARE REQUESTED
      &IF &INDEX EQ 2 &GOTO -S1
      &SPACE
      &PRINT GIMMIE FILENAME / LAST FIELD NAME
      &READ ARGS
      &IF &INDEX NE 2 &QUIT
      &IF &1 EQ QUIT &QUIT
-S1 FILEDEF RAMUTIL1 TAP2 RE FB LR 80 BL 800
      &STACK KT
      &STACK DUMP
      &STACK FILE &1
      &STACK PRINT &2 AND SAVE
      &STACK END
      &STACK FIN
ATTACH RAMIS ALL BR
      &STACK KT
RAMIS
DETACH RAMIS BR
FILEDEF RAMUTIL1 CLEAR
      &EXIT
```

Listing of Filename = COURSE1, Filetype = RAM

```
ERASE
FILE COURSE
END
INPUT
READ COURSE NODUP
RECORD 240
SET 1
FORM 4,4,1,10,16,3,4,3,3,5,5,2,5,5,5,5
FORM 4,4,3,4,3,3,5,5,3,3,4,3,3,5,5,3
FORM 4,5,2,2,4,3,3,2,6,5,4,5,10
ORDER  2,1,3,4,5,6,7,8,9,10,11,12,13,14,15,16
ORDER 17,18,19,20,21,22,23,24,25,26,27,28,29
ORDER 30,31,32,33,34,35,36,37,38,39,40,41,42
ORDER 43,44,45
FILE COURSE
END
```

Listing of Filename = COURSE2, Filetype = RAM

```
REVISE
READ COURSE NODUPS
RECORD 18
FORM 4,9,4,1
ORDER 1,53,54,55
KEY 1,53
INCLUDE 53
FILE COURSE
END
```

Listing of Filename = COURSE 3, Filetype = RAM

```
ERASE
FILE COURSE
END
INPUT
READ COURSE DUMP2
RECORD 240
SET 1
FORM 4,4,1,10,16,3,4,3,3,5,5,2,5,5,5,5
FORM 4,4,3,4,3,3,5,5,3,3,4,3,3,5,5,3
FORM 4,5,2,2,4,3,3,2,6,5,4,5,10
FORM 6,6,6,3,3,3,4,9,4,1
ORDER  1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
ORDER 17,18,19,20,21,22,23,24,25,26,27,28,29
ORDER 30,31,32,33,34,35,36,37,38,39,40,41,42
ORDER 43,44,45,46,47,48,49,50,51,52,53,54,55
FILE COURSE
END
```

Listing of Filename = FACIL1, Filetype = RAM

```
ERASE
FILE FACIL
END
INPUT
READ FACIL DUMP
RECORD 80
SET 1
FORM 4,4,5,3,3,1,6,5
ORDER 2,1,3,4,5,6,7,8
FILE FACIL
END
```

Listing of Filename = FACIL2, Filetype = RAM

```
ERASE
FILE FACIL
END
INPUT
READ FACIL DUMP
RECORD 80
SET 1
FORM 4,4,4,4,3,1,6,5
ORDER 1,2,3,4,5,6,7,8
FILE FACIL
END
```

Listing of Filename = INST1, Filetype = RAM

```
ERASE
FILE INST
END
INPUT
READ INST NODUP
RECORD 80
SET 1
FORM 9,16,2,5,4,4,4,4,4
ORDER 1,2,3,4,5,6,7,8,9
FILE INST
END
```

Listing of Filename = INST2, Filetype = RAM

```
ERASE
FILE INST
END
INPUT
READ INST DUMP2
RECORD 160
SET 1
FORM 9,16,2,5,4,4,4,4,4
FORM 4,4,4,4,4,4,4,4,4
FORM 4,4,4,4,4,4,4,4
ORDER 1,2,3,4,5,6,7,8,9
ORDER 10,11,12,13,14,15,16,17,18
ORDER 19,20,21,22,23,24,25,26
FILE INST
END
```

Listing of Filename = INDEX, Filetype = RUNPROC

```
DELETE
FILE RAMEXEC
1=OUT=INDEX,$
END
CATALOG INDEX
TABLE
HEADING
      RAMASTER DESCRIPTION FOR
      RAMIS FILE NAMED &1
      AT &TIME ON &DATE
FILE RAMASTER
LIST NAME AND ALIAS AND LEVEL AS 'LEV'
      AND TYPE AND LENGTH AS 'LEN'
      AND SEG AS 'SEG'
      AND LEVEL-TYPE AS 'SFLAG'
IF FILE IS &1
END
END CATALOG
```

SECTION VI

TRAINING PROCESS FLOW MODEL

PROGRAM DESCRIPTIONS

MODEL DESCRIPTION

The objectives of the Training Process Flow model system are to provide training management with the ability to assess the effects upon the training system of changes in demand, scheduling, and capacities, and to present the model output data so that effective decisions can be made for maximizing the utilization of training command resources. The Training Process Flow model system consists of the TPF model and those support programs and executive which interface with the DOTS data base, provide "what if" capability, control model execution, and provide terminal output capabilities. The overall flow of the TPF model system is shown in Figure VI-1. The support programs and their associated executives are described first, with the description of the TPF model last.

SUPPORT PROGRAMS. The TPF support programs fall into three categories: those necessary to interface the TPF model with the DOTS data base, those required to support the "what if" capability of the TPF model, and those required to support selective report capability for terminal operation.

TPF MODEL – SIMPLIFIED SYSTEM FLOW

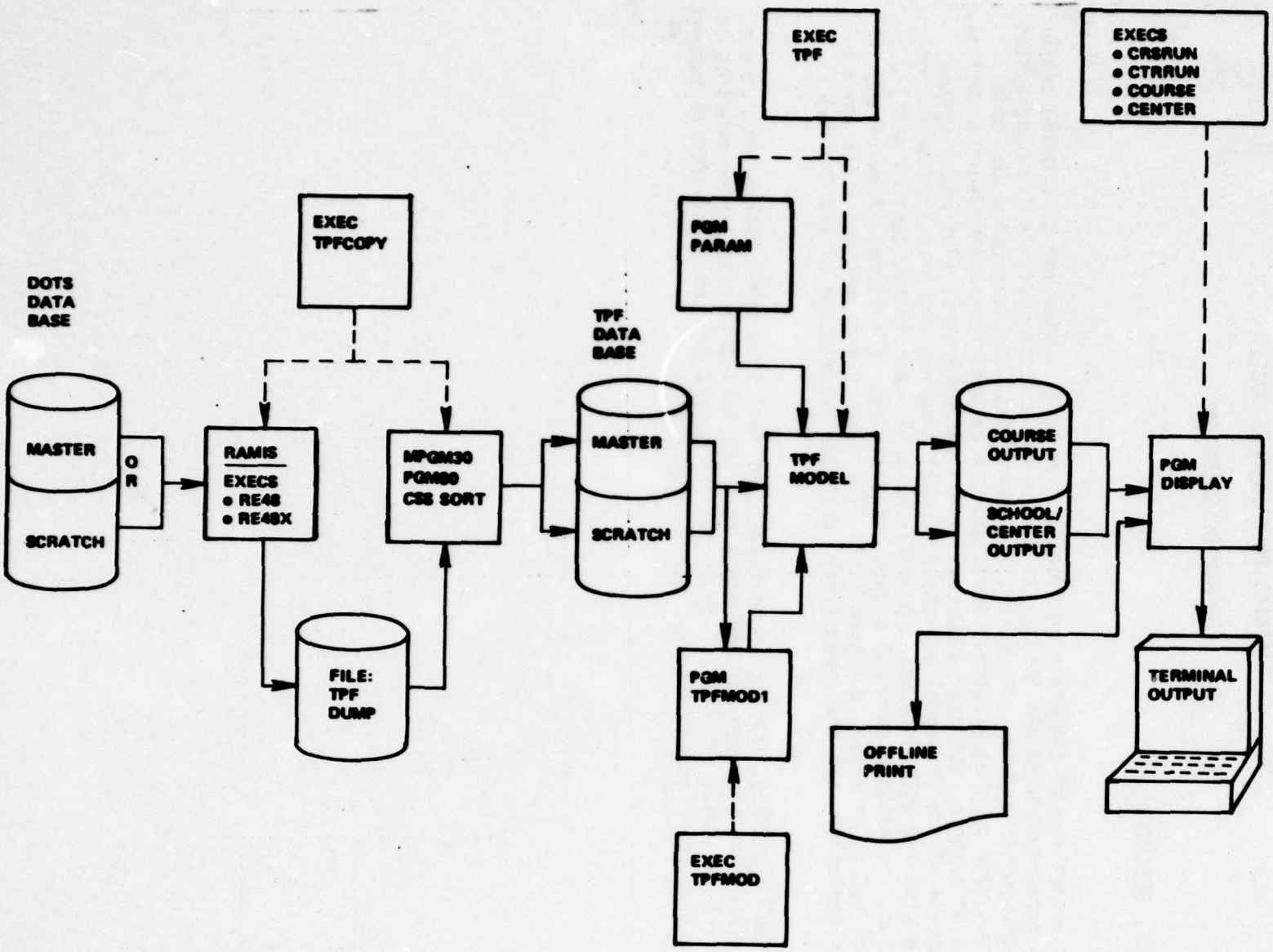


FIGURE VI-1

DATA BASE INTERFACE PROGRAMS

All data used for TPF model execution are stored in the DOTS DATA BASE (master or scratch) using the RAMIS system. A series of executives and programs is provided to facilitate unloading of the RAMIS data base, and recopying these data in a format compatible to the TPF model. The flow of these support programs and executives is shown in Figure VI-2. All data used by the DOTS system are stored in the DOTS Master data base. These data can be extracted or modified, and is stored in the DOTS Scratch data base. All procedures concerning the use and manipulation of these data bases are discussed in Section V.

Data from the DOTS data base must be copied into TPF compatible files prior to execution of the TPF model. The executive that controls this function is called TPFCOPY. When this executive is called, the parameter MASTER or SCRATCH may be provided to direct execution to the proper data base; if not, it will be asked for. The first step in the executive is to assure that the user's core is set to at least 320K Bytes, due to RAMIS requirements. If the core storage provided is not sufficient, an appropriate error message is displayed, core storage is set to the proper value, and the executive terminates. The executive must be restarted, as any set core operation causes a re-IPL of the user's system, which terminates any ongoing operation. The executive tests &INDEX for the number of parameters passed to the executive by the user. If no parameters are provided, the user is prompted to enter MASTER or SCRATCH. The executive then tests for the appropriate response, and will stay in this loop until one of these two keywords is provided.

Executive TPFCOPY then invokes RAMIS, which dumps file COURSE if MASTER was specified. A RAMIS procedure, either RE48 or RE48X controls this dump, and directs that only the first 36 fields (through UCAP) be dumped. INDEX0 will contain a zero if the dump is successful. If an error occurred, a message will be printed; otherwise, executive TPFSORT is called.

The data unloaded from RAMIS must be sorted into school code order prior to TPF execution. To accomplish this sort the unloaded RAMIS file dump, which consists of two 80 character records per course, must first be combined into a single 160 character record per course, sorted, and then returned to two 80 character card image records per course, compatible with the TPF input. Executive TPFSORT starts program MPGM3 to combine the records, then commands a CSS sort by school, by CDP, and then starts PGM80 to reformat the records to a TPF compatible format. The LISTF feature is used to track the execution of this executive.

The format of the data as it is unloaded from RAMIS into the file TPF DUMP is shown in Table VI-1. The format for the files TPF MASTER or TPF SCRATCH is shown in Figure VI-3, A and B. The data are stored as two 80 byte records per course.

TPF MODEL - DATA BASE INTERFACE

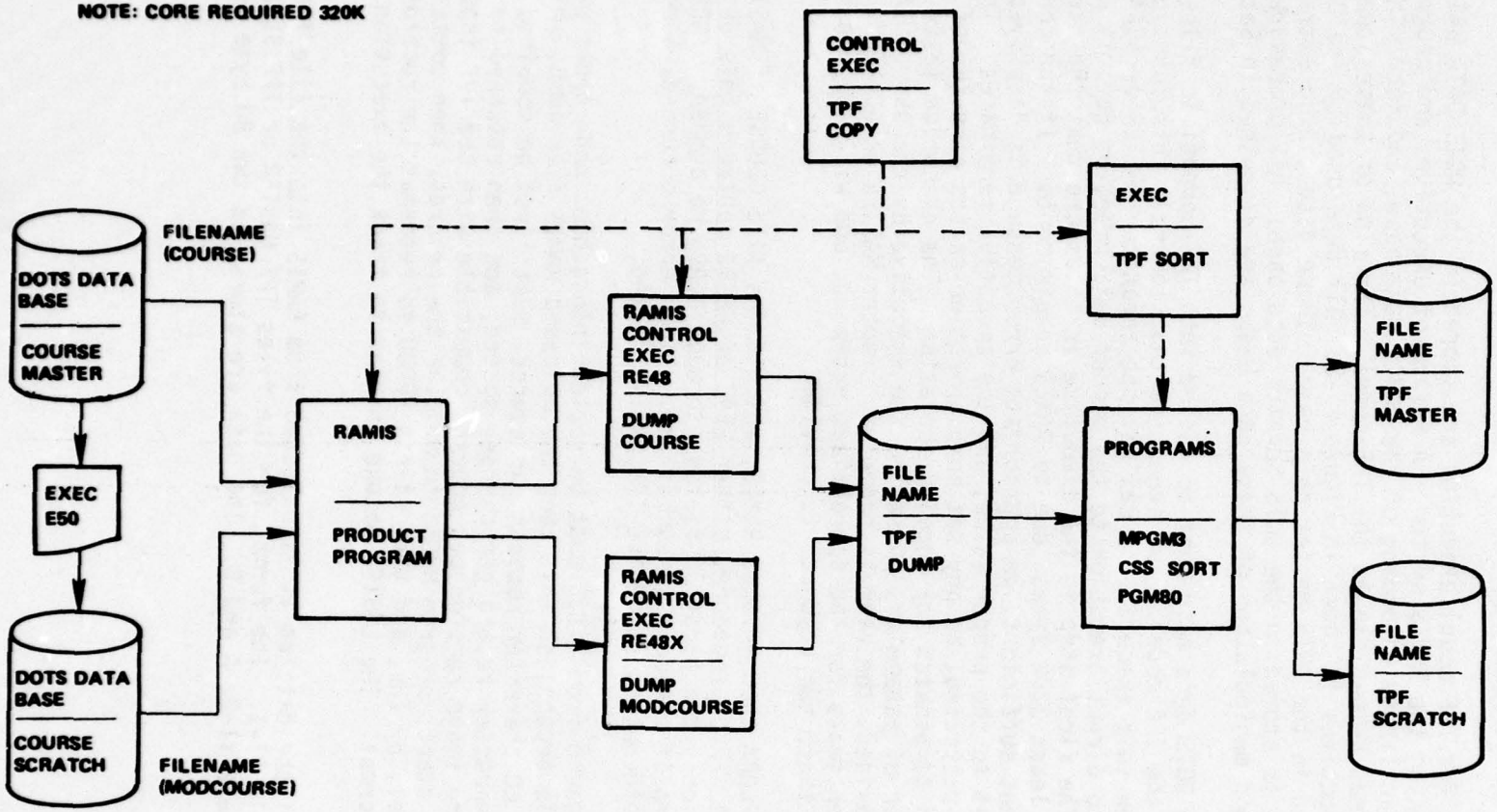


FIGURE VI-2

RAMIS DUMP FORMAT (MASTER OR SCRATCH)

± EBCDIC RECORD LENGTH 8Ø CHARACTERS

FIELD NAME	FORMAT		LENGTH
CDP	A 4		4
SCH	A 4		4
DEL	A 1		1
CTLG	A 1Ø		1Ø
CRS	A 16		16
CONV1	I 3		3
LEN1	F4 .1		4
CAP1	I 3		3
BCAP1	I 3		3
DMD1	I 5		5
BDMD1	I 5		5
OFST	I 2		2
PFAIL	F5 .3		5
NADIS	F5 .3		5
NOSHOW	F5 .3		5
SETB	F5 .3		5
		X	Ø
BKLG	I 4		4
JNO	A 4		4
CONV2	I 3		3
LEN2	F4 .1		4
CAP2	I 3		3
BCAP2	i 3		3
DMD2	I 5		5
BDMD2	I 5		5
CHWK2	I 3		3
CONV3	I 3		3
LEN3	F4 .1		4
CAP3	I 3		3
BCAP3	I 3		3
DMD3	I 5		5
BDMD3	I 5		5
CHWK3	I 3		3
PAOB	I 4		4
AFAIL	F5 .3		5
PRTY	I 2		2
CTYPE	A 2		2
PNEC	A 4		4
UCAP	I 3		3

TABLE VI-1

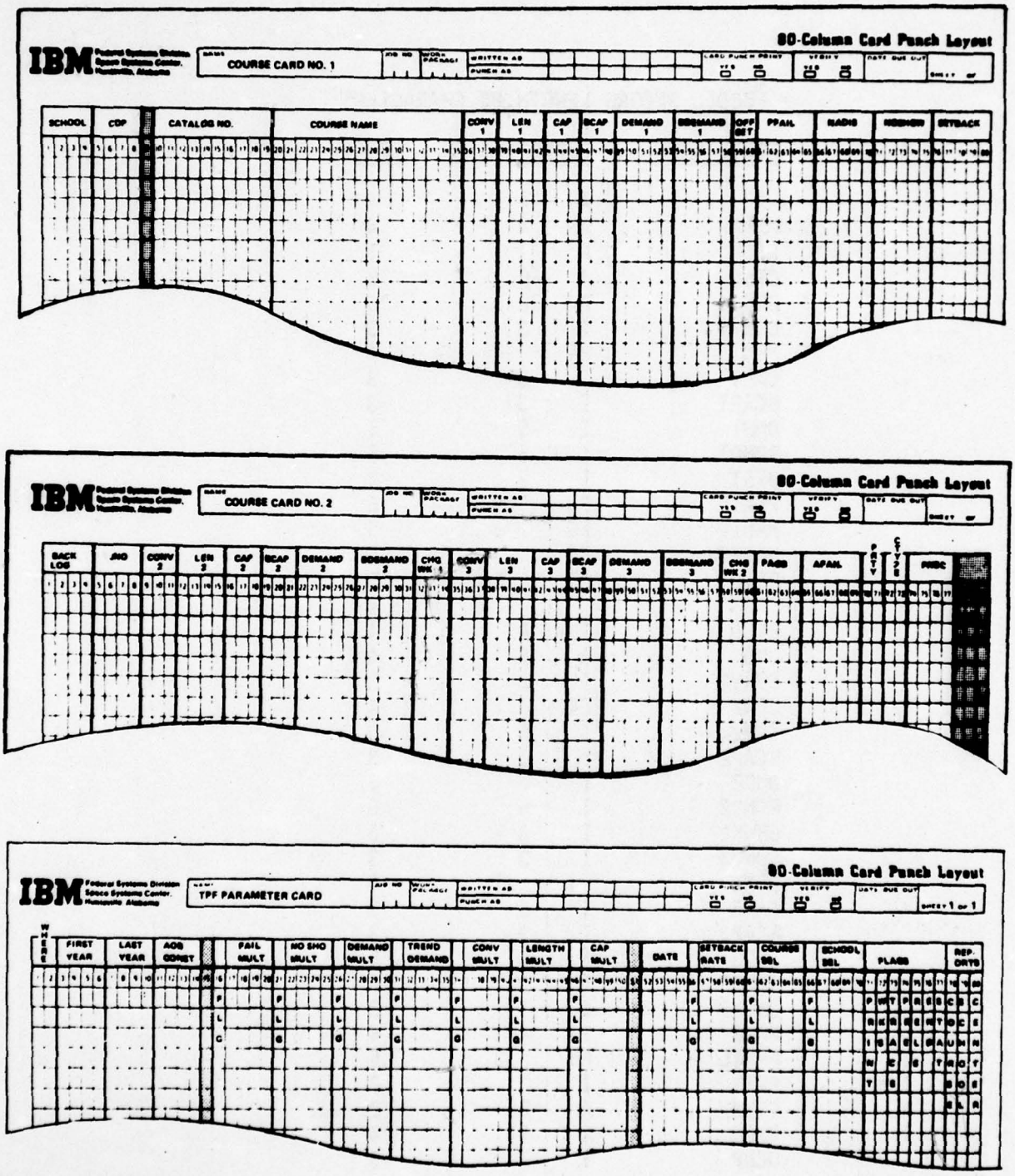

80-Column Card Punch Layout
IBM
COURSE CARD NO. 1
SCHOOL
CDP
CATALOG NO.
COURSE NAME
CONV 1
LEN 1
CAP 1
SCAP 1
DEMAND 1
DEMAND 1
OFF SET
PFAIL
RADIO
RESERV
SETBACK
80-Column Card Punch Layout
IBM
COURSE CARD NO. 2
BACK LOG
AVG
CONV 2
LEN 2
CAP 2
SCAP 2
DEMAND 2
DEMAND 2
CHG WK 1
CONV 3
LEN 3
CAP 3
SCAP 3
DEMAND 3
DEMAND 3
CHG WK 2
PASS
AFAIL
PRTY
CTYPE
MISC
80-Column Card Punch Layout
IBM
TPF PARAMETER CARD
WHERE
FIRST YEAR
LAST YEAR
AQG CONST
FAIL MULT
NO SHO MULT
DEMAND MULT
TREND DEMAND
CONV MULT
LENGTH MULT
CAP MULT
DATE
SETBACK RATE
COURSE SEL
SCHOOL SEL
PLACE
REPORTS

FIGURE VI-3

TPFCOPY EXEC

```
&TYPE OFF
&IF &CORE GE 320 &GOTO -OK
&PRINT CORE WAS NOT 320 K - USER MUST RESTART PROCEDURE
SET CORE 320
&EXIT
-OK &INDEX1 = 1
&IF &INDEX EQ 0 &GOTO -NONE
&IF &1 EQ MASTER &GOTO -RUNIT
&INDEX1 = 2
&IF &1 EQ SCRATCH &GOTO -RUNIT
-NONE &PRINT REPLY MASTER OR SCRATCH
&READ ARGS
&GOTO -OK
&EXIT
-RUNIT ATTACH RAMIS ALL BR
ERASE TPF DUMP
FILEDEF RAMUTIL1 DSK TPF DUMP LR 80 RE F
&STACK KT
&INDEX2 = 1
&IF &INDEX1 EQ 1 EXEC RAMISIN RE4H RAM
&IF &INDEX1 EQ 2 EXEC RAMISIN RE4HX RAM
&IF &INDEX0 NE 0 &INDEX2 = 10
DETACH RAMIS BR
&IF &INDEX2 LT 5 &GOTO -RAMOK
&STACK RT
&PRINT FILE NOT DUMPED- RAMIS ERROR
&EXIT
-RAMOK FILEDEF * CLEAR
&STACK RT
&IF &INDEX1 EQ 1 &PRINT TPF MASTER DUMPED BY RAMIS
&IF &INDEX1 EQ 2 &GOTO -SORTSCR
EXEC TPFSORT MASTER
&EXIT
-SORTSCR &IF &INDEX1 EQ 2 &PRINT TPF SCRATCH DUMPED BY RAMIS
EXEC TPFSORT SCRATCH
&EXIT
```

TABLE VI-2

TPFSORT EXEC

```
    &TYPE OFF
ATTACH TEMP20 AS 193 BRIEF
FILEDEF DUMP DSK TPF DUMP RE F LR 80 BL 80
FILEDEF DUMP2 DSK TPF WORK P RE F LR 160 BL 160
LOAD MPGM3 (CLEAR LIBE) PLILIB
START IHECMS 2
FILEDEF DUMP CLEAR
FILEDEF DUMP2 CLEAR
    &STACK 5 8 1 4
SORT TPF WORK TPF SORT (BRIEF #MODE=U)
LISTF TPF DUMP P (A)
LISTF TPF SORT U (A)
LISTF TPF WORK P (A)
ERASE TPF WORK P
FILEDEF SYSUT1 DSK TPF SORT U RE F BL 160 LR 160
FILEDEF SYSUT2 DSK TPF MASTER P RE F BL 80 LR 80
&IF &1 EQ SCRATCH FILEDEF SYSUT2 DSK TPF SCRATCH P RE F BL 80 LR 80
LOAD PGM80 (CLEAR LIBE) PLILIB
START (BRIEF)
&IF &1 EQ MASTER LISTF TPF MASTER P (A)
&IF &1 EQ SCRATCH LISTF TPF SCRATCH P (A)
FILEDEF SYSUT1 CLEAR
FILEDEF SYSUT2 CLEAR
DETACH 193 BRIEF
    &EXIT
```

RE48 RAM

```
USE
RAMDATA DATA U
END
DUMP
FILE COURSE
PRINT UCAP AND SAVE
END
```

RE48X RAM

```
USE
RAMDATA DATA U
END
DUMP
FILE MODCOURSE
PRINT UCAP AND SAVE
END
```

TABLE VI-3

```
MPGM3:  /* COMBINE //MORE CARDS INTO A SINGLE LRECL - 20/MAY/76 */
        PROC (PARMS) OPTIONS (MAIN) ;

DECLARE PARMS CHAR (100) VARYING,
        NEOF FIXED BIN (31,0) STATIC,
        IHESARC ENTRY ( FIXED BINARY (31,0) ) ;

DECLARE CARDS (240) CHAR (1) STATIC,
        CARD1 CHAR (80) BASED (PC1),
        CARD2 CHAR (80) BASED (PC2),
        CARD3 CHAR (80) BASED (PC3),
        BUF2 CHAR (160) BASED (PB),
        BUF3 CHAR (240) BASED (PB) ;

STARTERRUPP:
        GET STRING (PARMS) EDIT (NTOT) ( F(4) ) ;

        PB = ADDR (CARDS(1)) ;
        PC1 = ADDR (CARDS(1)) ;
        PC2 = ADDR (CARDS(81)) ;
        PC3 = ADDR (CARDS(161)) ;
        NR, NC = 0 ;

        OPEN FILE (DUMP) RECORD INPUT,
             FILE (DUMP2) RECORD OUTPUT ;
        ON ENDFILE (DUMP) GO TO EOJ ;

R1:     NEOF = 0 ;
        READ FILE (DUMP) INTO (CARD1) ;
             NC = NC + 1 ;
        NEOF = 4 ;
        READ FILE (DUMP) INTO (CARD2) ;
             NC = NC + 1 ;

        IF ( NTOT = 2 ) THEN GO TO W2 ;

        READ FILE (DUMP) INTO (CARD3) ;
             NC = NC + 1 ;

W3:          WRITE FILE (DUMP2) FROM (BUF3) ;
W3A:         NR = NR + 1 ;
                 GO TO R1 ;

W2:     WRITE FILE (DUMP2) FROM (BUF2) ;
                GO TO W3A ;

EOJ:    DISPLAY ( ' ' ) ;
        DISPLAY ( NC || ' : TOTAL INPUT DUMP RECORDS' ) ;
        DISPLAY ( NR || ' : TOTAL OUTPUT COMBINATION RECORDS' ) ;

        CLOSE FILE (DUMP), FILE (DUMP2) ;

        CALL IHESARC ( NEOF ) ;

             END MPGM3 ;
```

TABLE VI-4

PGM80 PLI

```
PGM80:  /* SCRUNCH 160 BYTE RECORDS TO 80 BYTE MULTIPLES - 7/20/76 */
        PROC OPTIONS (MAIN) ;

DECLARE C160 CHAR (160) BASED (P1),
        A160 (160) CHAR (1) STATIC,
        C1   CHAR (80) BASED (P1),
        C2   CHAR (80) BASED (P2),
        F1   CHAR (4) BASED (P1),
        F2   CHAR (4) BASED (P3),
        MOVE CHAR (4) STATIC ;

START:  P1 = ADDR ( A160(1) ) ;
        P2 = ADDR ( A160(81) ) ;
        P3 = ADDR ( A160(5) ) ;
        NC, NR = 0 ;

        OPEN FILE (SYSUT1) RECORD INPUT,
             FILE (SYSUT2) RECORD OUTPUT ;
        ON ENDFILE (SYSUT1) GO TO EOJ ;

R1:     READ FILE (SYSUT1) INTO (C160) ;
        NC = NC + 1 ;

             MOVE = F1 ;  F1 = F2 ;  F2 = MOVE ;

             WRITE FILE (SYSUT2) FROM (C1) ;
             WRITE FILE (SYSUT2) FROM (C2) ;
             NR = NR + 2 ;

                  GO TO R1 ;

EOJ:    CLOSE FILE (SYSUT1), FILE (SYSUT2) ;

        DISPLAY ( NC || ' : INPUT RECORDS PROCESSED' ) ;
        DISPLAY ( NR || ' : OUTPUT RECORDS PROCESSED' ) ;

             END PGM80 ;
```

TABLE VI-5

"WHAT IF" SUPPORT PROGRAMS

The TPF model user normally will use the TPF system first to examine a school or center under normal, or baseline, conditions, and then to modify certain input parameters and observe the changes to the outputs that occur. Three approaches are available to the user to modify these input data. See Figure VI-4. The first is to use the RAMIS facilities to modify the baseline data and create a DOTS scratch data base. The use of RAMIS to accomplish this is discussed in Section II of this report. The second method is to execute program TPFMOD1, which allows the user to make modifications to individual courses for study, or to insert now or upcoming courses in order to observe their effects on the overall system. Finally, program PARAM can be used to make modifications to a certain type of data for all courses; i.e., reducing the demand for all courses by 10%.

COURSE MODIFICATIONS. Executive TPFMOD, in conjunction with program TPFMOD1, is used to control the "what if" capability of the TPF system for modifications to individual courses or the study or inclusion of new courses to the center.

Executive TPFMOD. Executive TPFMOD is quite straight forward, and its flow is shown in Figure VI-5. This executive sets up the program to access either the TPF MASTER or TPF SCRATCH data base created by copying and reformatting the data stored in the DOTS data base. A third file, TPF VFY, shown in Figure VI-4. is available to the TPF model for verification purposes, but cannot be accessed by TPFMOD1. Three features of executive TPFMOD should be explained. First, &INDEX contains the number of parameters passed the executive during the call (or parameter read), and &1 ... &10 contain the parameters. It is important to test that there are parameters before testing the value of the parameters as the executive will ABEND on a value test if none are present. Secondly, &INDEX0 contains the condition code of the last program run. In this case, it is used to test the condition code of the STOP command in program TPFMOD1, which will be 0 if the user desires to save the data created, and 999 if it is not to be saved. Finally, program TPFMOD1 overrides the normal error checking for input/output errors of the System/370, and this error checking must be reset by the SET SPIE ON command to allow normal processing to continue. Failure to reset this SPIE may cause subsequent programs to loop during execution of I/O commands, resulting in excessive consumption of computer time and charges.

Program TPFMOD1. Program TPFMOD1 is actually a load module obtained by combining main program TPFMOD with subroutines COMND, CONVRT, MSG, DECODE, and SHOW. The FORTRAN program TPFMOD is discussed first, with each subroutine discussed separately.

Program TPFMOD has two purposes. The first is to allow the user to access specific courses from the TPF data base (master or scratch), make alterations to any of the data elements, and to store the modified course data for TPF model execution. The second purpose is to allow the user to create new courses, prompted by the program, and again store them for TPF execution. Program TPFMOD (main) controls the sequence of execution to accomplish this. This control program can be broken into three parts: The first part controls the "COPY" option, the second controls the "NEW" option, while the third section controls the "MOD" option. Several features are consistent throughout this main program and the various subroutines. First, all data set input uses the direct access method of FORTRAN read. This is to allow the user to skip through the files without

TPF MODEL - "WHAT IF" AND OUTPUT DISPLAY

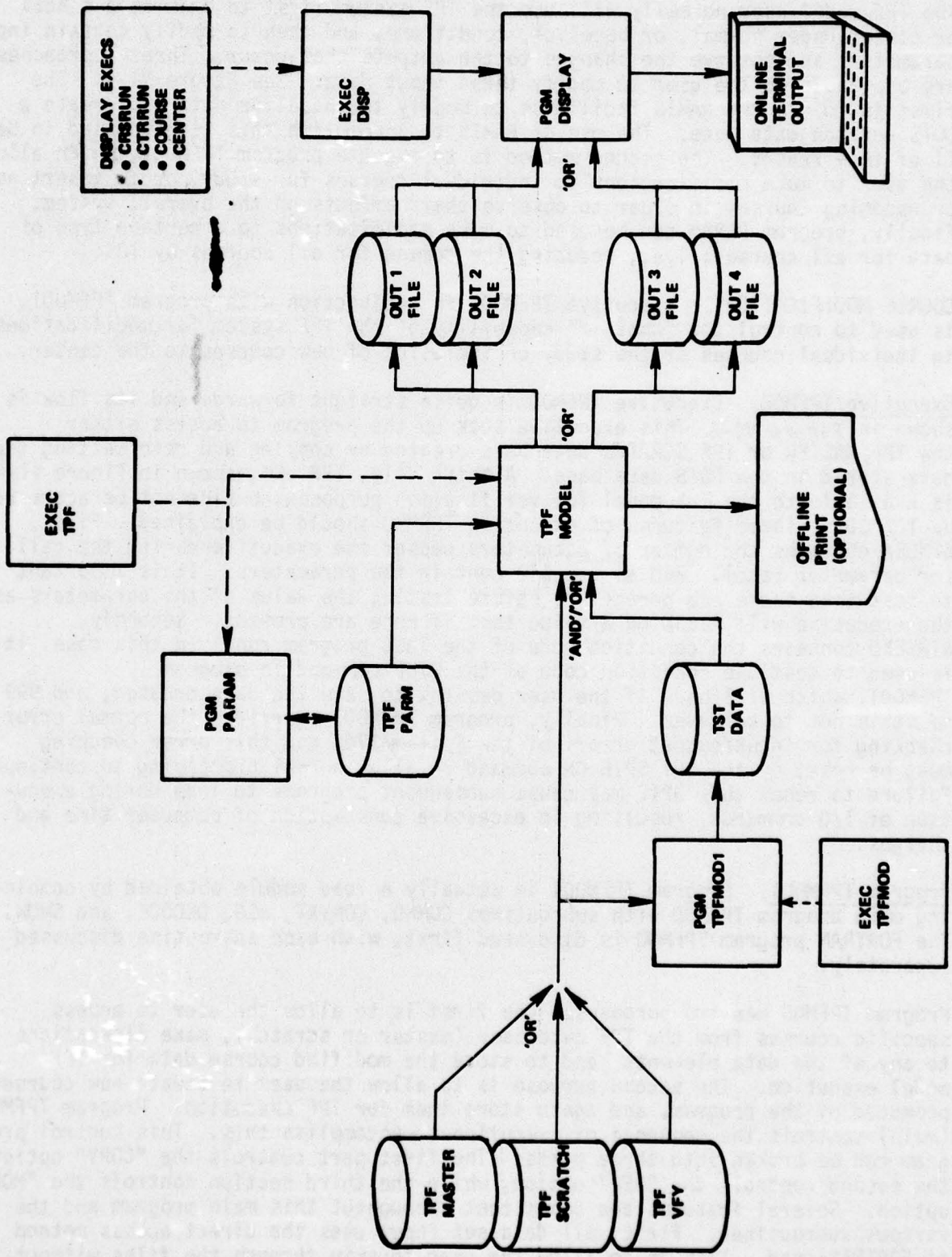


FIGURE VI-4

the need to open and close them on each read. A CALL ERRSET command is declared at the beginning of the program to override error checking during this I/O operation. Should the user specify a course CDP number not in file, inline subroutine RANGE will be invoked to recover the error and allow processing to continue.

Secondly, most terminal I/O uses the 10A1 format. This is to allow the user to intermix alpha, numeric, and decimal characters on the same read operation and to allow entry of a variable number of parameters during the same read. Subroutine CONVRT is used to change the results of this read back to internally usable data. Finally, during any terminal read operation, terms such as HELP, QUIT, SHOW, DONE, etc., may be entered, and subroutine MSG will provide the proper action.

The COPY option allows the user to copy all data for a particular course into core storage for subsequent modification and finally execution by the TPF model. The course is called by the CDP number, and either file TPF MASTER or TPF SCRATCH is searched for that course. The course data are stored in the order they are accessed rather than by CDP number. This is to allow evaluation of a course under several conditions during a single model run, hence the same course can be copied several times. However, due to the pointer method of storage, the sequence of operation must be COPY, MODIFY, COPY, MODIFY...as it is impossible to modify course data after a new course has been read in.

The NEW option allows the user to create new courses in the file. The user is prompted for the proper data and format for only those data elements required for TPF execution. This prompting is accomplished through a DO loop, which repetitively calls subroutine DECODE, handing off the proper parameter. The course counter is incremented once a new course has been inserted in the file.

The MOD option allows the user to modify data for a course inserted in the file, whether through the COPY or NEW option. The user is prompted to enter a field name. Subroutine DECODE then validates the field name and prompts for the data in the proper format. Subroutine DECODE also accepts the field name, followed by a blank and then the data to speed up the operation by an experienced operator. The field names in this program are identical to the field names used in the DOTS data base, and may be displayed by entering HELP and asking for the field names.

The wrapup of this program allows the user to bypass storage of the file created if errors occurred, and also supports the HELP option. The HELP option allows the user to display the field names or commands available, without reference to an operation manual. Processing returns to the point at which HELP was entered upon completion of this option.

Subroutine DECODE has three purposes. The first is to support the NEW option by prompting for proper field data. The second is to support the MOD option, by verifying field names, and prompting for and verifying the field data. Finally, the subroutine supports the STAT command, and will display the current value of any field for the course in process. Communication with the main program, or other calling subroutines, is accomplished through parameters passed with the subroutine call for control and return codes, and through COMMON storage for data elements. DECODE is called in the form:

CALL DECODE (I, N, RESP, IN, IN2)

where: All data elements are of the form INTEGER*2

I contains the valid field number if known (usually through the new option), a zero if it is not known, or a 9999 if the STAT option is in effect (from subroutine SHOW).

N is a pointer at the data in storage for the course being processed.

RESP contains the return code (0 for valid operation, 99 for an invalid operation, or the subroutine MSG return code if the operator keyed in a valid message during execution.

IN and IN2 are 5 element arrays containing the data to be converted. IN contains the field name, while IN2 contains the data.

Subroutine DECODE contains the type of data (integer, alpha, or decimal) for each field in an array called TYPE, and the limiting value for each field in an array called SIZE. This allows error checking of data elements and allows prompting of the user if no data are entered with the field name or an error occurs. During execution of subroutine DECODE, if any data entered are invalid or not recognized, subroutine MSG is called to ascertain if a valid message was entered, and if so, subroutine DECODE terminates, using the message number as a return code.

Subroutine CONVRT is a FORTRAN subroutine called by TPFMOD MAIN and subroutine DECODE, as well as other programs elsewhere in the TPF system. The purpose of CONVRT is to convert data read into storage in the A format to a data form compatible with internal processing. The A format is used in the TPF system for terminal inputs to allow intermixing of data, messages, and field names, allow blank delimited input, to allow intermixing of integer and decimal data, and to eliminate the need for right or left justification for data input. This subroutine is called in the form:

CALL CONVRT (IN, ANS, CODE, DECMAL)

where: IN is in the form INTEGER*2 and is a 5 element array containing the data to be converted.

ANS is in the form INTEGER*4 and contains the integer conversion.

CODE is in the form INTEGER*2 and contains the return code for the subroutine (0 for integer conversion, 5 for decimal conversion, 10 for invalid conversion).

DECMAL is in the form REAL*4 and contains both the real and fractional portion of the answer for a decimal conversion.

Basic operation of this subroutine is quite simple. Each digit of the number to be converted was read into storage using the A1 format. Thus, the number 1 is stored in an INTEGER*2 location as hexidecimal F140, rather than the desired 0001, and is thus not valid for processing. To convert each digit to a processable number the value F000 is first subtracted from the number, in this case

leaving 0140. The number is then divided by 256, which causes a shift right 2, leaving 0001. An equivalence statement is used, making the work area for this operation an INTEGER*4, to prevent underflows. Traps are provided to recognize multiple decimals. The five individual values decoded are then combined into a single decimal or integer value.

Subroutine MSG is a short FORTRAN subroutine used to decode one of 9 valid text messages passed from the calling program or subroutine. It is called in the form:

CALL MSG (IN, CODE)

where: IN is in the form INTEGER*2 and is a 5 element array containing the message to be decoded.

CODE is in the form INTEGER*2 and contains one of the following return codes:

QUIT(1), DONE(2), HELP(3), STAT(4), SHOW(5),
COPY(6), NEW(7), ALL(8), MOD(9), NOT RECOGNIZED(99).

Subroutine SHOW is a FORTRAN subroutine to provide display of all elements for the course in process. This subroutine is called in the form:

CALL SHOW (N)

where: N is in the form INTEGER*2 and contains a pointer to the course in the data matrix in process.

The user is allowed to specify a specific field to be displayed, or call for all fields by means of a CR when prompted. The format of the data displayed is similar to that used for the TPF model. However, the student backlog figure used in the TPF model is found during model initialization, while the figure used here is calculated, with the possibility that in a few cases the backlog figure may differ slightly. No return code is provided for in this subroutine, hence messages cannot be decoded. This subroutine also has the BLOCK DATA subroutine attached, which accomplishes initialization of all data in COMMON storage prior to program execution.

Subroutine COMND is an assembly language subroutine whose execution causes the screen of the Hazeltine display to be cleared. It has no effect on most other terminals. It is called in the form:

CALL COMND (I, J)

where: I and J are INTEGER*2 variables, I must contain a 6 to cause a clear operations and J contains the return code (0 for valid operation, 1 or 2 for error).

The return code is not tested. This subroutine is used by most FORTRAN and PL/I programs in the TPF system.

ACROSS THE BOARD MODIFICATIONS. Program PARAM, in conjunction with executive

TPF, is used to support "what if" capability of the TPF system for "across the board" modifications to input data. This is accomplished through manipulation of data set TPF PARM which is an input to the TPF model, and through the use of FILEDEF commands in the executive to direct the input/output for the TPF model.

Executive TPF. Executive TPF controls execution of both program PARAM and the TPF model. The executive requests which data sets are to be used as input and sets up the necessary flags in &INDEX1 and &INDEX2. An option exists to enter the word SAVE as the second parameter when the input file is requested, and if present, will set a flag in &INDEX3. Normally center and school output data for the TPF model will be written in OUT3 FILE, and detailed course data will be written in OUT4 FILE, overwriting data already written there. If the SAVE option is specified, output will be directed to OUT1 FILE for center and school data, and OUT2 FILE for course data. Also, the TST DATA file, created by program TPFMOD1 can be concatenated with any other input data set at this time.

Program PARAM. Program PARAM is a PL/I program called by the executive TPF, and is used to modify the contents of data set TPF PARM. This data set contains a single 80 byte record that is used as input data by the TPF model to control execution. The format of this data set is shown in Figure VI-3 C. This method of control for the TPF model was chosen to facilitate off-line or batch execution, as well as on-line terminal applications. The program reads the current record from data set TPF PARM, displays the active elements, allows the user to make changes, and then rewrites the record. The first four elements (input, first year, last year, and AOB constant) are always active and their values will be displayed. The next 10 fields are active only if a number greater than zero appears in the flag column. Finally, the last 10 fields contain flags, and an indication will be displayed if their value is non-zero. Use of this parameter data are discussed in the TPF LOGIC DESIGN section. The program runs in a manner similar to the other support programs in this section, and accepts most commands, such as HELP,QUIT, and SHOW as the other program. Internally, most of the field location and data assignment tasks are done using the substring feature of the PL/1 language. This is to allow the variable display format desired for the active fields, and assist in decoding and locating fields for data entry. The COMND subroutine is used to provide the clear display feature at the beginning of the program and PL/1 subroutine IHESARC is used to pass a return code to bypass execution of the TPF model if the user so directs. Finally, the date of the run is acquired through the PL/I date feature, and is passed to the TPF model on the PARM card.

FLOW - EXECUTIVE TPFMOD

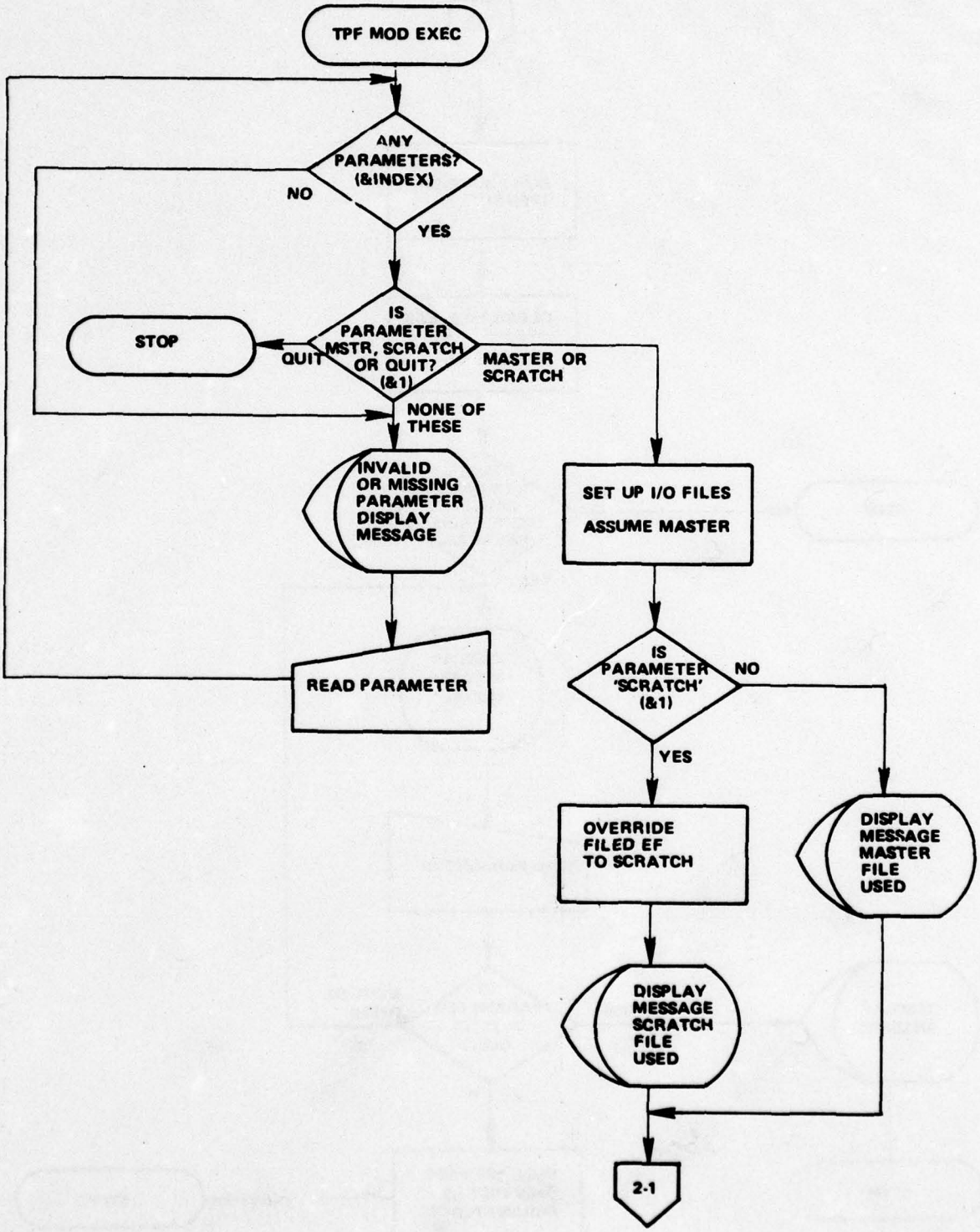


FIGURE VI-5

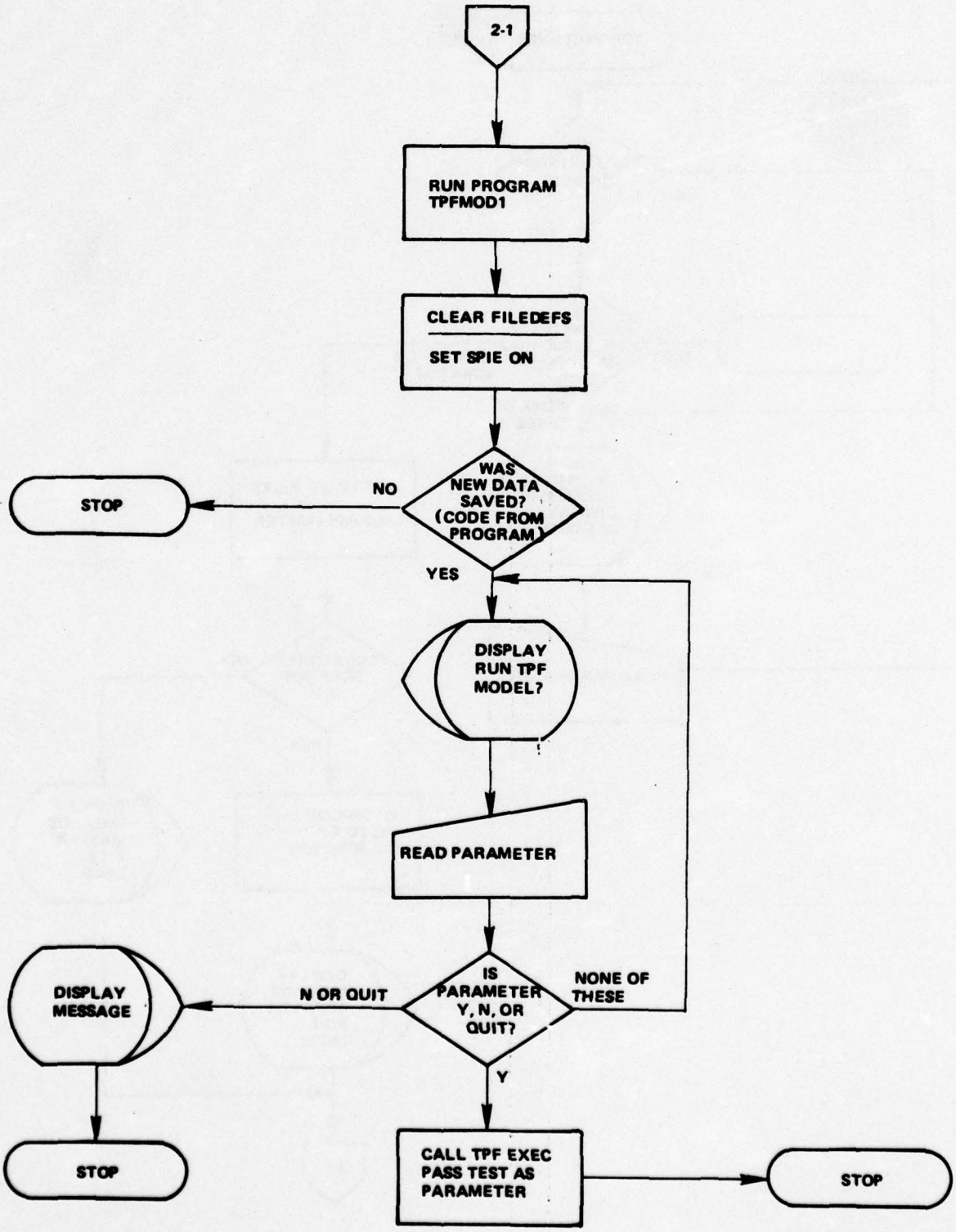


FIGURE VI-5 (Continued)

LISTING - EXECUTIVE TPFMOD

```
&TYPE OFF
-FRST &IF &INDEX EQ Ø &GOTO -NONE
&IF &1 EQ MASTER &GOTO -OK
&IF &1 EQ SCRATCH &GOTO -OK
&IF &1 EQ QUIT &EXIT
-NONE &PRINT REPLY MASTER OR SCRATCH (INPUT FILE)
&READ ARGS
&GOTO -FRST
-OK FILEDEF * CLEAR
FILEDEF 3 DSK TPF MASTER
FILEDEF 4 DSK TST DATA
&IF &1 NE SCRATCH &GOTO -MASTER
FILEDEF 3 DSK TPF SCRATCH
&PRINT SCRATCH DATA BASE USED AS MOD INPUT
&GOTO -RUNIT
-MASTER &PRINT MASTER DATA BASE USED AS MOD INPUT
-RUNIT TPFMOD1
FILEDEF * CLEAR
SET SPIE ON
&IF &INDEXØ GE 999 &GOTO -QUIT
-READ &PRINT RUN TPF MODEL? (Y OR N)
&READ ARGS
&IF &1 EQ Y &GOTO -MODEL
&IF &1 EQ N &GOTO -SKIP
&IF &1 EQ QUIT &GOTO -SKIP
&GOTO -READ
-MODEL EXEC TPF TEST
-SKIP &EXIT
-QUIT &PRINT NOT SAVED AT OPERATORS REQUEST
&EXIT
```

TABLE VI-6

08/14/75 INPUT LISTING -TPFMOD-

LISTING - PROGRAM TPFMOD

FORTRAN MODULE (LIST,NAMSQ,SOURCSEQ)

```
CARD NO  ****                          CONTENTS                                  ****
      1  C TPF INPUT CREATE/MODIFY PROGRAM 13JUN75                                00000100
      2  C SUBROUTINES CALLED-MSG, CONVRT, DECODE, COMND                          00000200
      3        IMPLICIT INTEGER*2 (S)                                             00000300
      4        INTEGER*4 S1,S16,S35,S36                                           00000400
      5        REAL*4 S5,S11,S12,S13,S14,S18,S25,S31,S32                          00000500
      6        REAL*8 S2A,S3A,S3B                                                 00000600
      7        INTEGER*2 YES/'Y'/, NO/'N'/                                        00000700
      8        COMMON /A/ S1(144),S2A(144),S3A(144),S4(144),S5(144),S6(144)       00000800
      9        COMMON /A/ S7(144),S8(144),S9(144),S10(144),S11(144),S12(144)      00000900
     10        COMMON /A/ S13(144),S14(144),S15(144),S16(144),S17(144),S18(144)   00001000
     11        COMMON /A/ S19(144),S20(144),S21(144),S22(144),S23(144),S24(144)   00001100
     12        COMMON /A/ S25(144),S26(144),S27(144),S28(144),S29(144),S30(144)   00001200
     13        COMMON /A/ S31(144),S32(144),S33(144),S34(144),S35(144),S36(144)   00001300
     14        COMMON /A/ S2B (144),S3B(144)                                      00001400
     15        INTEGER*2 IN(5) /5*0/, IN2 (5) /5*0/, CODE, NJ/6/,JN               00001500
     16        COMMON NGR                                                         00001600
     17        INTEGER*4 NEW /'NEW '/, QUIT /'QUIT'/, DONE /'DONE'/, TEXT, TCDP   00001700
     18        DEFINE FILE 3 (500,80,E,J3)                                        00001800
     19        EXTERNAL RANGE                                                     00001900
     20        CALL ERRSET (232,100,-1,1,RANGE)                                   00002000
     21        CALL COMND (NJ,JN)                                                 00002100
     22        N = 0                                                              00002200
     23        NGR = 0                                                            00002300
     24   1000 WRITE (6,1)                                                        00002400
     25      1 FORMAT (/' TPF COURSE CREATE/MODIFY PROGRAM'/)                     00002500
     26      2 WRITE (6,3)                                                       00002600
     27      3 FORMAT (' +ENTER ''NEW'' OR ''COPY'' '/)                          00002700
     28      4 READ (5,5) IN                                                     00002800
     29      5 FORMAT (10A1)                                                     00002900
     30        CALL MSG (IN, CODE)                                               00003000
     31        IF (CODE .LE. 2) GO TO 999                                        00003100
     32        IF (CODE .EQ. 6) GO TO 100                                        00003200
     33        IF (CODE .EQ. 7) GO TO 200                                        00003300
     34        IF (CODE .EQ. 8) GO TO 300                                        00003400
     35        IF (CODE .EQ. 9) GO TO 400                                        00003500
     36        WRITE (6,6) IN                                                    00003600
     37      6 FORMAT (1X, 5A1, ' +INVALID- RETRY'/)                             00003700
     38        GO TO 2                                                           00003800
     39  C ***** COPY ***** COPY EXISTING COURSE FROM FILE 3                     00003900
     40    100 WRITE (6,101)                                                     00004000
     41    101 FORMAT (/ ' +COPY EXISTING COURSE- ENTER CDP')                    00004100
     42    102 READ (5,103) TEXT                                                 00004200
     43    103 FORMAT (A4)                                                       00004300
     44        IF (TEXT .EQ. QUIT) GO TO 999                                     00004400
     45        IF (TEXT .EQ. DONE) GO TO 1000                                    00004500
     46        IF (TEXT .EQ. NEW)  GO TO 200                                     00004600
     47        IF (TEXT .EQ. ALL)  GO TO 300                                     00004700
     48        IF (TEXT .EQ. HELP) CALL HELP1                                    00004800
     49        IF (TEXT .EQ. HELP) GO TO 100                                     00004900
     50  C HERE IF DIDNT RECOGNIZE MESSAGE- SEE IF IS IN CDP FILE                00005000
     51        DO 110 I = 1, 400, 2                                              00005100
     52        READ (3'I, 103, ERR=115) TCDP                                     00005200
     53        IF (NGR .GT. 0) GO TO 115                                         00005300
     54        IF (TCDP .EQ. TEXT) GO TO 120                                     00005400
     55    110 CONTINUE                                                          00005500
     56  C HERE IF ERROR OR NOT IN FILE                                          00005600
     57    115 NGR = 0                                                           00005700
     58        WRITE (6,116) TEXT                                                00005800
     59    116 FORMAT (1X, A4, ' +NOT IN FILE- RETRY OR ENTER DONE'/)            00005900
     60        GO TO 102                                                         00006000
     61  C HERE IF FOUND CDP                                                     00006100
     62    120 CONTINUE                                                          00006200
     63        N = N + 1                                                         00006300
     64        J = I                                                             00006400
     65        READ (3'J,188,ERR=115) S1(N),S2A(N),S2B(N),S3A(N),S3B(N),S4(N),S5(00006500
     66       *N), S6(N),S7(N),S8(N),S9(N),S10(N),S11(N),S12(N),S13(N),S14(N),   00006600
     67       * S15(N),S16(N),S17(N),S18(N),S19(N),S20(N),S21(N),S22(N),S23(N),  00006700
     68       * S24(N),S25(N),S26(N),S27(N),S28(N),S29(N),S30(N),S31(N),         00006800
     69       * S32(N),S33(N),S34(N),S35(N),S36(N)                               00006900
     70    188 FORMAT (A4,A8,A2,2A8,I3,F3.1,2I3,2I5,I2,4F4.3,I4,A4,2X,/,4X,      00007000
     71       * 2(I3,F3.1,2I3,2I5,I3),2F4.1,2A2,2A4)                            00007100
     72    121 WRITE (6,122) TEXT                                                00007200
     73    122 FORMAT (/ ' CDP ', A4, ' +COPIED- ENTER ''DONE'' OR ''MOD'' ')    00007300
     74    123 READ (5,5) IN,IN2                                                 00007400
     75        CALL MSG (IN, CODE)                                               00007500
     76        GO TO (999, 1000, 124, 126, 125, 100, 200, 300, 400), CODE        00007600
     77        WRITE (6,6) IN                                                    00007700
     78        GO TO 123                                                         00007800
     79    124 CALL HELP1                                                        00007900
     80        GO TO 121                                                         00008000
     81    125 CALL SHOW (N)                                                     00008100
     82        GO TO 121                                                         00008200
     83    126 I = 9999                                                          00008300
     84        CALL DECODE (I,N,CODE,IN,IN2)                                     00008400
     85        IF (CODE .EQ. 0) GO TO 121                                        00008500
     86        WRITE (6,6) IN2                                                   00008600
     87        GO TO 123                                                         00008700
     88  C **** NEW **** CREATE NEW COURSE- PROMPT FOR ENTRIES                   00008800
     89    200 CALL COMND (NJ,JN)                                                00008900
     90    201 WRITE (6,202)                                                     00009000
     91    202 FORMAT (/' CREATE NEW COURSE'/)                                   00009100
     92        N1 = N + 1                                                        00009200
     93        I = 1                                                             00009300
     94        CALL DECODE (I,N1,CODE,IN,IN2)                                    00009400
     95        IF (CODE .EQ. 0) GO TO 210                                        00009500
     96        GO TO (999,1000,204,203,203,100,200,300), CODE                    00009600
     97    203 WRITE (6,7)                                                       00009700
     98      7 FORMAT (' +INVALID- RETRY'/)                                      00009800
     99        GO TO 201                                                         00009900
    100    204 CALL HELP1                                                        00010000
    101        GO TO 201                                                         00010100
    102  C HERE IF FOUND CDP- NOW INCRIMENT COUNTER & CREATE COURSE               00010200
    103    210 WRITE (6,211)                                                     00010300
    104    211 FORMAT (/' ENTER FOLLOWING DATA TO CREATE COURSE'/)               00010400
    105        N = N + 1                                                         00010500
    106        DO 230 I = 3, 16                                                  00010600
    107        IF (I .EQ.10) GO TO 230                                           00010700
    108        IF (I .EQ.14) GO TO 230                                           00010800
    109    212 I1 = I                                                            00010900
```

08/14/75 INPUT LISTING -TPFMOD-

```
CARD NO  ****                          CONTENTS                          ****
   110          IF (I .EQ. 16) II = 36                                          00011000
   111          CALL DECODE (II,N,CODE,IN,IN2)                                  00011100
   112          IF (CODE .EQ. 0) GO TO 230                                      00011200
   113          GO TO (999,1000,221,212,222,100,200,300,400), CODE              00011300
   114          WRITE (6,6) IN                                                  00011400
   115          GO TO 212                                                       00011500
   116        221 CALL HELP1                                                    00011600
   117          GO TO 212                                                       00011700
   118        222 CALL SHOW (N)                                                 00011800
   119          GO TO 212                                                       00011900
   120        230 WRITE (6,229)                                                 00012000
   121        229 FORMAT (' ')                                                  00012100
   122      C END PROMPTED COURSE CREATE                                        00012200
   123        231 WRITE (6,232)                                                 00012300
   124        232 FORMAT (/' KEY DATA ENTERED FOR CREATE- ENTER ''MOD'' OR ''DONE''  00012400
   125           *')                                                            00012500
   126        236 READ (5,5) IN,IN2                                             00012600
   127            CALL MSG (IN,CODE)                                          00012700
   128            GO TO (999,1000,233,235,234,100,200,300,400), CODE          00012800
   129            WRITE (6,6) IN                                              00012900
   130            GO TO 236                                                   00013000
   131        233 CALL HELP1                                                  00013100
   132            GO TO 231                                                   00013200
   133        235 I = 9999                                                    00013300
   134            CALL DECODE (I,N,CODE,IN,IN2)                               00013400
   135            IF (CODE .EQ. 0) GO TO 231                                  00013500
   136            WRITE (6,6) IN2                                             00013600
   137            GO TO 236                                                   00013700
   138        234 CALL SHOW (N)                                               00013800
   139            GO TO 231                                                   00013900
   140      C **** ALL **** HERE IF COPY ALL                                    00014000
   141        300 WRITE (6,301)                                               00014100
   142        301 FORMAT (/' ''ALL'' OPTION NOT SUPPORTED THIS PROGRAM' /)    00014200
   143            GO TO 1000                                                  00014300
   144      C **** MOD **** HERE IF MOD CALLED                                  00014400
   145        400 IF (N .EQ. 0) GO TO 412                                     00014500
   146            WRITE (6,401) B1 (N)                                        00014600
   147        401 FORMAT (/' MODIFY DATA FOR CDP ',A4)                        00014700
   148        402 WRITE (6,403)                                               00014800
   149        403 FORMAT (/' +ENTER FIELD TO BE CHANGED')                     00014900
   150        404 READ (5,5) IN,IN2                                           00015000
   151            I = 0                                                       00015100
   152            CALL DECODE (I,N,CODE,IN,IN2)                               00015200
   153            IF (CODE .EQ. 0) GO TO 402                                  00015300
   154            GO TO (999,1000,410,402,411,100,200,300,400), CODE          00015400
   155            WRITE (6,6) IN                                              00015500
   156            GO TO 402                                                   00015600
   157        410 CALL HELP1                                                  00015700
   158            GO TO 402                                                   00015800
   159        411 CALL SHOW (N)                                               00015900
   160            GO TO 402                                                   00016000
   161        412 WRITE (6,413)                                               00016100
   162        413 FORMAT (' MOD INVALID- NO COURSE IN FILE'/)                  00016200
   163            GO TO 1000                                                  00016300
   164      C **** WRAPUP ****                                                00016400
   165        999 WRITE (6,501)                                               00016500
   166        501 FORMAT (/' SAVE NEW DATA BASE? (Y OR N)')                   00016600
   167            READ (5,5) IN                                               00016700
   168            IF (IN (1) .EQ. YES) GO TO 510                              00016800
   169            IF (IN (1) .EQ. NO) GO TO 1999                              00016900
   170            CALL MSG (IN,CODE)                                          00017000
   171            GO TO (1999,1999,520,520,520,100,200,300,400),CODE          00017100
   172        520 WRITE (6,6) IN                                              00017200
   173            GO TO 999                                                   00017300
   174        510 NSAVE = N                                                   00017400
   175            DO 515 N = 1, NSAVE                                         00017500
   176        515 WRITE (4,516) B1(N),B2A(N),B2B(N),B3A(N),B3B(N),B4(N),B5(   00017600
   177           *N), B6(N),B7(N),B8(N),B9(N),B10(N),B11(N),B12(N),B13(N),B14(N),  00017700
   178           * B15(N),B16(N),B1(N),B17(N),B18(N),B19(N),B20(N),B21(N),B22(N),B23  00017800
   179           * (N),B24(N),B25(N),B26(N),B27(N),B28(N),B29(N),B30(N),B31(N),  00017900
   180           * B32(N),B33(N),B34(N),B35(N),B36(N)                           00018000
   181        516 FORMAT (A4,A8,A2,2A8,I3,F3.0,2I3,2I5,I2,4F4.3,I4,A4,2X,/,A4,  00018100
   182           * 2(I3,F3.0,2I3,2I5,I3),2F4.1,2A2,2A4)                        00018200
   183            STOP                                                        00018300
   184       1999 STOP 999                                                    00018400
   185            END                                                         00018500
   186            SUBROUTINE HELP1                                            00018600
   187            INTEGER*4 TEXT,QUIT/'QUIT'/,SKIP/'SKIP'/,FIEL/'FIEL'/,CODE/'CODE'/  00018700
   188            INTEGER*4 DONE /'DONE'/                                     00018800
   189          1 WRITE (6,3)                                                 00018900
   190          3 FORMAT (/' +HELP CALLED- ENTER FIELD, CODES, OR SKIP')      00019000
   191            READ (5,5) TEXT                                             00019100
   192          5 FORMAT (A4)                                                 00019200
   193            IF (TEXT .EQ. QUIT) GO TO 999                               00019300
   194            IF (TEXT .EQ. DONE) GO TO 999                               00019400
   195            IF (TEXT .EQ. SKIP) GO TO 999                               00019500
   196            IF (TEXT .EQ. FIEL) GO TO 100                               00019600
   197            IF (TEXT .EQ. CODE) GO TO 200                               00019700
   198            WRITE (6,10) TEXT                                           00019800
   199         10 FORMAT (1X, A4, ' INVALID- RETRY')                          00019900
   200            GO TO 1                                                     00020000
   201        100 WRITE (6,101)                                               00020100
   202        101 FORMAT (//' TPF FIELDS ARE:'/                               00020200
   203           *' CDP    CONV1  LEN1   CAP1   BCAP1  DMD1   BDMD1  OFST'/   00020300
   204           *' CTLG   CONV2  LEN2   CAP2   BCAP2  DMD2   BDMD2  CHWK1'/  00020400
   205           *' CRS    CONV3  LEN3   CAP3   BCAP3  DMD3   BDMD3  CHWK2'/  00020500
   206           *' SCH    PFAIL  NADIS  NOSHO  SETB   BKLG'/                 00020600
   207           *' JNO    PNEC   PAOB   AFAIL  PRTY   CTYPE'//)              00020700
   208            GO TO 999                                                   00020800
   209        200 WRITE (6,201)                                               00020900
   210        201 FORMAT (//' VALID CODES ARE:'/                              00021000
   211           *' QUIT  DONE  HELP  STAT  SHOW  COPY  NEW  ALL  MOD'//)     00021100
   212            GO TO 999                                                   00021200
   213        999 RETURN                                                      00021300
   214            END                                                         00021400
   215            SUBROUTINE RANGE (KK,LL)                                    00021500
   216            COMMON MSR                                                  00021600
   217            MSR = 10                                                    00021700
   218            KK = 0                                                      00021800
   219            LL = 232                                                    00021900
   220            RETURN                                                      00022000
   221            END                                                         00022100
```

TABLE VI-7 (Continued)

08/14/75 INPUT LISTING -DECODE-

LISTING - SUBROUTINE DECODE

FORTRAN MODULE (LIST,NAMSQ,SOURCSEQ)

```
CARD NO  ****                          CONTENTS                                ****
      1  C DECODE SUBROUTINE FOR TPF 'WHAT IF' 22MAY75                          00000100
      2        SUBROUTINE DECODE (I,N,RESP,IN,IN2)                              00000200
      3        IMPLICIT INTEGER*2 (8)                                           00000300
      4        INTEGER*4 $1,$16,$35,$36                                         00000400
      5        REAL*4 $5,$11,$12,$13,$14,$18,$25,$31,$32                        00000500
      6        REAL*8 $2A,$3A,$3B                                               00000600
      7  C                                                                      00000700
      8        COMMON /A/ $1(144),$2A(144),$3A(144),$4(144),$5(144),$6(144)     00000800
      9        COMMON /A/ $7(144),$8(144),$9(144),$10(144),$11(144),$12(144)    00000900
     10        COMMON /A/ $13(144),$14(144),$15(144),$16(144),$17(144),$18(144) 00001000
     11        COMMON /A/ $19(144),$20(144),$21(144),$22(144),$23(144),$24(144) 00001100
     12        COMMON /A/ $25(144),$26(144),$27(144),$28(144),$29(144),$30(144) 00001200
     13        COMMON /A/ $31(144),$32(144),$33(144),$34(144),$35(144),$36(144) 00001300
     14        COMMON /A/ $2B (144),$3B(144)                                    00001400
     15        INTEGER*2 NUM (5) /5*0/, CODE, TYPE (36), DEC, INT, ALPHA        00001500
     16        DATA TYPE /3*'A', 7*'I', 4*'F', 'I', 'A', 15*'I','F', 4*'A'/     00001600
     17        DATA ALPHA /'A'/, INT /'I'/, DEC /'F'/                           00001700
     18        REAL*8 ALPHA1, ALPHA2                                            00001800
     19        INTEGER*4 ANS, A4 (2),QUIT/'QUIT'/,DONE/'DONE'/,HELP/'HELP'/,SHOW/00001900
     20       *'SHOW'/                                                          00002000
     21        INTEGER*2 IN (5),IN2(5),F (5,36), CDP (5), CTLG (5), CRS (5)     00002100
     22        INTEGER*2 CONV1 (5), LEN1 (5), CAP1 (5), BCAP1 (5), DMD1 (5)     00002200
     23        INTEGER*2 BDMD1 (5), OFST (5), PFAIL (5), NADIS (5), NOSMO (5)   00002300
     24        INTEGER*2 SETB (5), BKLG (5), JNO (5), CONV2 (5), LEN2 (5)       00002400
     25        INTEGER*2 CAP2 (5), BCAP2 (5), DMD2 (5), BDMD2 (5), CHWK2 (5)    00002500
     26        INTEGER*2 CONV3 (5), LEN3 (5), CAP3 (5), BCAP3 (5), DMD3 (5)     00002600
     27        INTEGER*2 BDMD3 (5), CHWK3 (5), PAOB (5), AFAIL (5), PRTY (5)    00002700
     28        INTEGER*2 CTYPE (5), PNEC (5), SCH (5), BLANK /' '/              00002800
     29        INTEGER*2 RESP, SIZE (36), A2 (4), NA2 (4)                       00002900
     30        DATA SIZE /4, 10, 16, 3, 4, 2*3, 2*5, 2, 4*5, 2*4, 3, 4,         00003000
     31       * 2*3, 2*5, 2*3, 4, 2*3, 2*5, 3, 4, 5, 2*2, 2*4/                  00003100
     32        EQUIVALENCE (A2,ALPHA1), (A4,ALPHA1), (NA2,ALPHA2)               00003200
     33        EQUIVALENCE (CDP, F(1,1)), (CTLG, F(1,2)), (CRS, F(1,3))         00003300
     34        EQUIVALENCE (CONV1, F(1,4)), (LEN1, F(1,5)), (CAP1, F(1,6))      00003400
     35        EQUIVALENCE (BCAP1, F(1,7)), (DMD1, F(1,8)), (BDMD1, F(1,9))     00003500
     36        EQUIVALENCE (OFST, F(1,10)), (PFAIL, F(1,11)), (NADIS, F(1,12))  00003600
     37        EQUIVALENCE (NOSMO, F(1,13)), (SETB, F(1,14)), (BKLG, F(1,15))   00003700
     38        EQUIVALENCE (JNO, F(1,16)), (CONV2, F(1,17)), (LEN2, F(1,18))    00003800
     39        EQUIVALENCE (CAP2, F(1,19)), (BCAP2, F(1,20)), (DMD2, F(1,21))   00003900
     40        EQUIVALENCE (BDMD2, F(1,22)), (CHWK2, F(1,23)), (CONV3, F(1,24))  00004000
     41        EQUIVALENCE (LEN3, F(1,25)), (CAP3, F(1,26)), (BCAP3, F(1,27))   00004100
     42        EQUIVALENCE (DMD3, F(1,28)), (BDMD3, F(1,29)), (CHWK3, F(1,30))  00004200
     43        EQUIVALENCE (PAOB, F(1,31)), (AFAIL, F(1,32)), (PRTY, F(1,33))   00004300
     44        EQUIVALENCE (CTYPE, F(1,34)), (PNEC, F(1,35)), (SCH, F(1,36))    00004400
     45        DATA CDP/'C','D','P',' ',' '/,CTLG/'C','T','L','G',' '/          00004500
     46        DATA CRS/'C','R','S',' ',' '/,CONV1/'C','O','N','V','1'/         00004600
     47        DATA LEN1/'L','E','N','1',' '/,CAP1/'C','A','P','1',' '/         00004700
     48        DATA BCAP1/'B','C','A','P','1'/,DMD1/'D','M','D','1',' '/        00004800
     49        DATA BDMD1/'B','D','M','D','1'/,OFST/'O','F','S','T',' '/        00004900
     50        DATA PFAIL/'P','F','A','I','L'/,NADIS/'N','A','D','I','S'/       00005000
     51        DATA NOSMO/'N','O','S','M','O'/,SETB/'S','E','T','B',' '/        00005100
     52        DATA SCH/'S','C','H',' ',' '/,BKLG/'B','K','L','G',' '/          00005200
     53        DATA JNO/'J','N','O',' ',' '/,CONV2/'C','O','N','V','2'/         00005300
     54        DATA LEN2/'L','E','N','2',' '/,CAP2/'C','A','P','2',' '/         00005400
     55        DATA BCAP2/'B','C','A','P','2'/,DMD2/'D','M','D','2',' '/        00005500
     56        DATA BDMD2/'B','D','M','D','2'/,CHWK2/'C','H','W','K','2'/       00005600
     57        DATA CONV3/'C','O','N','V','3'/,LEN3/'L','E','N','3',' '/        00005700
     58        DATA CAP3/'C','A','P','3',' '/,BCAP3/'B','C','A','P','3'/        00005800
     59        DATA DMD3/'D','M','D','3',' '/,BDMD3/'B','D','M','D','3'/        00005900
     60        DATA CHWK3/'C','H','W','K','3'/,PAOB/'P','A','O','B',' '/        00006000
     61        DATA AFAIL/'A','F','A','I','L'/,PRTY/'P','R','T','Y',' '/        00006100
     62        DATA CTYPE/'C','T','Y','P','E'/,PNEC/'P','N','E','C',' '/        00006200
     63        RESP = 0                                                         00006300
     64        ISAVE = 1                                                        00006400
     65        NEWFL = 1                                                        00006500
     66        IF (I .EQ. 9999) I = 0                                           00006600
     67  C IF I HAS VALUE ALREADY KNOW CODE- SKIP                               00006700
     68        IF (I .GT. 0) GO TO 100                                          00006800
     69  C DONT KNOW IF FIELD OR MSG- SEE IF MSG                                00006900
     70        NEWFL = 0                                                        00007000
     71        CALL MSG (IN,CODE)                                               00007100
     72        IF (CODE .GE. 98) GO TO 905                                      00007200
     73  C HERE IF MSG- SEE IF STAT                                             00007300
     74        IF (CODE .EQ. 4) GO TO 900                                       00007400
     75  C NOT STAT- RETURN FROM SUBR WITH MSG CODE                             00007500
     76        RESP = CODE                                                      00007600
     77        GO TO 999                                                        00007700
     78  C **** STAT **** - LOAD IN2 INTO IN                                    00007800
     79    900 DO 902 J = 1,5                                                   00007900
     80    902 IN(J) = IN2(J)                                                   00008000
     81        ISAVE = 9999                                                     00008100
     82  C SEARCH OUT FIELD NUMBER                                              00008200
     83    905 DO 910 J = 1,36                                                  00008300
     84        DO 911 K = 1,5                                                   00008400
     85        IF (IN (K) .NE. F (K,J)) GO TO 910                               00008500
     86    911 CONTINUE                                                         00008600
     87        I = J                                                            00008700
     88        IF (ISAVE .EQ. 9999) GO TO 200                                   00008800
     89        GO TO 950                                                        00008900
     90    910 CONTINUE                                                         00009000
     91        RESP = 99                                                        00009100
     92        GO TO 999                                                        00009200
     93  C FIELD VALID- TEST IF DATA IN SECOND WORD                             00009300
     94    950 DO 960 J = 1,5                                                   00009400
     95        IF (IN2(J) .NE. BLANK) GO TO 970                                 00009500
     96    960 CONTINUE                                                         00009600
     97  C HERE IF BLANK FIELD- GO AND READ INPUT DATA                          00009700
     98        GO TO 100                                                        00009800
     99  C HERE IF SECOND DATA FIELD- BLANK DELIMIT                             00009900
    100    970 K = 0                                                            00010000
    101        DO 980 J = 1,5                                                   00010100
    102        NUM (J) = BLANK                                                  00010200
    103        IF (IN2(J) .EQ. BLANK) GO TO 980                                 00010300
    104        K = K + 1                                                        00010400
    105        NUM (K) = IN2 (J)                                                00010500
    106    980 CONTINUE                                                         00010600
    107        GO TO 985                                                        00010700
    108  C **** PROMPT DATA ENTRY ****                                          00010800
    109    100 WRITE (6,101) (F (J,I),J = 1,5), TYPE (I), SIZE (I)              00010900
```

08/14/75 INPUT LISTING -DECODE-

CARD NO	**** CONTENTS	****
110	101 FORMAT (' *ENTER ',5A1, 2X, '(',A1, I2, ')')	00011000
111	IF (TYPE (I) .EQ. ALPHA) GO TO 110	00011100
112	READ (5,102) NUM, IN2	00011200
113	102 FORMAT (10A1)	00011300
114	905 CALL CONVRT (NUM,ANS,CODE,DECMAL)	00011400
115	IF (CODE .EQ.10) GO TO 105	00011500
116	IF (CODE .EQ. 0) DECMAL = ANS	00011600
117	GO TO 118	00011700
118	105 CALL MSG (NUM, CODE)	00011800
119	IF (CODE .LT. 98) GO TO 108	00011900
120	C HERE IF MSG DID NOT RECOGNIZE TEXT (CODE=99), PRINT ERROR	00012000
121	IF (NEWFL .GT. 0) GO TO 1106	00012100
122	DO 1107 J = 1,5	00012200
123	1107 IN (J) = NUM (J)	00012300
124	GO TO 905	00012400
125	1106 WRITE (6,106) NUM	00012500
126	106 FORMAT (1X, 5A1, ' *INVALID- RETRY'/)	00012600
127	GO TO 100	00012700
128	108 RESP = CODE	00012800
129	IF (CODE .EQ. 4) GO TO 900	00012900
130	GO TO 999	00013000
131	110 READ (5,111) ALPHA1, ALPHA2	00013100
132	111 FORMAT (2A8)	00013200
133	RESP = 0	00013300
134	IF (A4 (1) .EQ. QUIT) RESP = 1	00013400
135	IF (A4 (1) .EQ. DONE) RESP = 2	00013500
136	IF (A4 (1) .EQ. HELP) RESP = 3	00013600
137	IF (A4 (1) .EQ. SHOW) RESP = 5	00013700
138	IF (RESP .NE. 0) GO TO 999	00013800
139	IF (SIZE (I) .EQ. 4) GO TO 112	00013900
140	IF (SIZE (I) .EQ. 2) GO TO 113	00014000
141	IF (SIZE (I) .EQ. 10) GO TO 114	00014100
142	IF (SIZE (I) .EQ. 16) WRITE (6,1111) ALPHA1, ALPHA2	00014200
143	1111 FORMAT (1X,2A8)	00014300
144	GO TO 121	00014400
145	112 WRITE (6,115) A4 (1)	00014500
146	115 FORMAT (1X,A4)	00014600
147	GO TO 121	00014700
148	113 WRITE (6,116) A2 (1)	00014800
149	116 FORMAT (1X,A2)	00014900
150	GO TO 121	00015000
151	114 WRITE (6,117) ALPHA1,NA2 (1)	00015100
152	117 FORMAT (1X, A8, A2)	00015200
153	GO TO 121	00015300
154	118 IF (TYPE (I) .EQ. DEC) GO TO 1118	00015400
155	C HERE IF INTEGER	00015500
156	ITEST = 10 ** SIZE (I)	00015600
157	IF (ANS .GE. ITEST) GO TO 1119	00015700
158	IF (ANS .GT. 32760) GO TO 1119	00015800
159	WRITE (6,119) ANS	00015900
160	GO TO 121	00016000
161	1119 WRITE (6,1120) ANS	00016100
162	1120 FORMAT (1X,I9,' TOO LARGE- RETRY'/)	00016200
163	GO TO 100	00016300
164	1118 WRITE (6,120) DECMAL	00016400
165	119 FORMAT (1X, I5)	00016500
166	120 FORMAT (F9.3)	00016600
167	121 CONTINUE	00016700
168	C **** WRITE TO CRS ****	00016800
169	IF (I .GT. 10) GO TO 710	00016900
170	GO TO (1,2,3,4,5,6,7,8,9,10),I	00017000
171	1 S1(N) = A4(1)	00017100
172	GO TO 999	00017200
173	2 S2A(N) = ALPHA1	00017300
174	S2B(N) = NA2 (1)	00017400
175	GO TO 999	00017500
176	3 S3A(N) = ALPHA1	00017600
177	S3B(N) = ALPHA2	00017700
178	GO TO 999	00017800
179	4 S4 (N) = ANS	00017900
180	GO TO 999	00018000
181	5 S5 (N) = ANS	00018100
182	GO TO 999	00018200
183	6 S6 (N) = ANS	00018300
184	GO TO 999	00018400
185	7 S7(N) = ANS	00018500
186	GO TO 999	00018600
187	8 S8 (N) = ANS	00018700
188	GO TO 999	00018800
189	9 S9(N) = ANS	00018900
190	GO TO 999	00019000
191	10 S10(N) = ANS	00019100
192	GO TO 999	00019200
193	710 IF (I .GT. 20) GO TO 720	00019300
194	II = I - 10	00019400
195	GO TO (11,12,13,14,15,16,17,18,19,20),II	00019500
196	11 S11(N) = DECMAL	00019600
197	GO TO 999	00019700
198	12 S12(N) = DECMAL	00019800
199	GO TO 999	00019900
200	13 S13(N) = DECMAL	00020000
201	GO TO 999	00020100
202	14 S14(N) = DECMAL	00020200
203	GO TO 999	00020300
204	15 S15 (N) = ANS	00020400
205	GO TO 999	00020500
206	16 S16(N) = A4(1)	00020600
207	GO TO 999	00020700
208	17 S17 (N) = ANS	00020800
209	GO TO 999	00020900
210	18 S18(N) = ANS	00021000
211	GO TO 999	00021100
212	19 S19(N) = ANS	00021200
213	GO TO 999	00021300
214	20 S20(N) = ANS	00021400
215	GO TO 999	00021500
216	720 II = I - 20	00021600
217	GO TO (21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36),II	00021700
218	21 S21(N) = ANS	00021800
219	GO TO 999	00021900
220	22 S22(N) = ANS	00022000
221	GO TO 999	00022100

TABLE VI-8 (Continued)

08/14/75 INPUT LISTING -DECODE-

CARD NO	**** CONTENTS	****
222	23 823(N) = ANS	00022200
223	GO TO 999	00022300
224	24 824 (N) = ANS	00022400
225	GO TO 999	00022500
226	25 825(N) = ANS	00022600
227	GO TO 999	00022700
228	26 826(N) = ANS	00022800
229	GO TO 999	00022900
230	27 827(N) = ANS	00023000
231	GO TO 999	00023100
232	28 828(N) = ANS	00023200
233	GO TO 999	00023300
234	29 829(N) = ANS	00023400
235	GO TO 999	00023500
236	30 830(N) = ANS	00023600
237	GO TO 999	00023700
238	31 831(N) = ANS	00023800
239	GO TO 999	00023900
240	32 832 (N) = DECMAL	00024000
241	GO TO 999	00024100
242	33 833(N) = ANS	00024200
243	GO TO 999	00024300
244	34 834(N) = A2(1)	00024400
245	GO TO 999	00024500
246	35 835(N) = A4(1)	00024600
247	GO TO 999	00024700
248	36 836(N) = A4(1)	00024800
249	GO TO 999	00024900
250	C **** STAT **** SHOW (STAT EQUATIONS - HERE IF HAVE FOUND FIELD	00025000
251	200 IF (I .GT. 10) GO TO 210	00025100
252	GO TO (301,302,303,304,305,306,307,308,309,310),I	00025200
253	301 A4(1) = S1(N)	00025300
254	GO TO 404	00025400
255	302 ALPHA1 = S2A(N)	00025500
256	NA2(1) = S2B (N)	00025600
257	GO TO 410	00025700
258	303 ALPHA1 = S3A(N)	00025800
259	ALPHA2 = S3B(N)	00025900
260	GO TO 416	00026000
261	304 ANS = S4 (N)	00026100
262	GO TO 400	00026200
263	305 DECMAL = S5(N)	00026300
264	GO TO 401	00026400
265	306 ANS = S6(N)	00026500
266	GO TO 400	00026600
267	307 ANS = S7(N)	00026700
268	GO TO 400	00026800
269	308 ANS = S8(N)	00026900
270	GO TO 400	00027000
271	309 ANS = S9(N)	00027100
272	GO TO 400	00027200
273	310 ANS = S10(N)	00027300
274	GO TO 400	00027400
275	210 IF (I .GT. 20) GO TO 220	00027500
276	II = I - 10	00027600
277	GO TO (311,312,313,314,315,316,317,318,319,320),II	00027700
278	311 DECMAL = S11(N)	00027800
279	GO TO 401	00027900
280	312 DECMAL = S12(N)	00028000
281	GO TO 401	00028100
282	313 DECMAL = S13(N)	00028200
283	GO TO 401	00028300
284	314 DECMAL = S14(N)	00028400
285	GO TO 401	00028500
286	315 ANS = S15(N)	00028600
287	GO TO 400	00028700
288	316 A4(1) = S16(N)	00028800
289	GO TO 404	00028900
290	317 ANS = S17(N)	00029000
291	GO TO 400	00029100
292	318 DECMAL = S18(N)	00029200
293	GO TO 401	00029300
294	319 ANS = S19(N)	00029400
295	GO TO 400	00029500
296	320 ANS = S20(N)	00029600
297	GO TO 400	00029700
298	220 II = I - 20	00029800
299	GO TO (321,322,323,324,325,326,327,328,329,330,331,332,333,334,335	00029900
300	*,336),II	00030000
301	321 ANS = S21(N)	00030100
302	GO TO 400	00030200
303	322 ANS = S22(N)	00030300
304	GO TO 400	00030400
305	323 ANS = S23(N)	00030500
306	GO TO 400	00030600
307	324 ANS = S24(N)	00030700
308	GO TO 400	00030800
309	325 DECMAL = S25(N)	00030900
310	GO TO 401	00031000
311	326 ANS = S26(N)	00031100
312	GO TO 400	00031200
313	327 ANS = S27(N)	00031300
314	GO TO 400	00031400
315	328 ANS = S28(N)	00031500
316	GO TO 400	00031600
317	329 ANS = S29(N)	00031700
318	GO TO 400	00031800
319	330 ANS = S30(N)	00031900
320	GO TO 400	00032000
321	331 DECMAL = S31(N)	00032100
322	GO TO 401	00032200
323	332 DECMAL = S32(N)	00032300
324	GO TO 401	00032400
325	333 A2(1) = S33(N)	00032500
326	GO TO 402	00032600
327	334 A2(1) = S34(N)	00032700
328	GO TO 402	00032800
329	335 A4(1) = S35(N)	00032900
330	GO TO 404	00033000
331	336 A4(1) = S36(N)	00033100
332	GO TO 404	00033200
333	400 WRITE (6,500) (F(J,1),J = 1,5),ANS	00033300

TABLE NO-B (continued)

08/14/75 INPUT LISTING -DECODE-

CARD NO	****	CONTENTS	****
334	500	FORMAT (/2X,9A1,1X,I5/)	00033400
335		GO TO 999	00033500
336	401	WRITE (6,501) (F (J,I),J = 1,5),DECNAL	00033600
337	501	FORMAT (/2X,9A1,F9.3/)	00033700
338		GO TO 999	00033800
339	404	WRITE (6,504) (F (J,I),J = 1,5),A4(1)	00033900
340	504	FORMAT (/2X,9A1,2X,A4/)	00034000
341		GO TO 999	00034100
342	402	WRITE (6,502) (F (J,I),J = 1,5),A2(1)	00034200
343	502	FORMAT (/2X,9A1,A2,/)	00034300
344		GO TO 999	00034400
345	410	WRITE (6,510) (F (J,I),J = 1,5),ALPHA1,NA2(1)	00034500
346	510	FORMAT (/2X,9A1,A8,A2/)	00034600
347		GO TO 999	00034700
348	416	WRITE (6,516) (F (J,I),J = 1,5),ALPHA1,ALPHA2	00034800
349	516	FORMAT (/2X,9A1,A8,A8/)	00034900
350	999	RETURN	00035000
351		END	00035100

TABLE VI-8 (Continued)

08/14/75 INPUT LISTING -CONVRT-

LISTING - SUBROUTINE CONVRT

FORTRAN MODULE (LIST,NAMSO,SOURCSBO)

```
CARD NO  ****  CONTENTS                                                                  ****
     1         C CONVERSION SUBROUTINE - UPDATE 28MAY75                                  00000100
     2         C CONVERSION OF 9 DIGET CHARACTER TO NUM                                  00000200
     3         C RETURN CODE 0 FOR VALID CONVERT, 10 FOR INVALID                         00000300
     4               SUBROUTINE CONVRT (IN, ANS, CODE, DECMAL)                           00000400
     5               INTEGER*2 IN (9) , WORK (10), CODE                                  00000500
     6               INTEGER*4 UPLIM, LOLIM, BLANK, HOLD (9), TEST, OUT (9), ANS,POINT   00000600
     7               EQUIVALENCE (HOLD, WORK)                                            00000700
     8               DATA WORK/1000/,TEST/ZF000/,UPLIM/ZF040/,LOLIM/ZF040/,BLANK/Z4040/  00000800
     9               DATA POINT /Z4040/                                                  00000900
    10               ANS = 0                                                             00001000
    11               PLG = 0                                                             00001100
    12               FRACT = 0                                                           00001200
    13               J = 1                                                               00001300
    14               DO 35 I = 1,9                                                       00001400
    15               K = 1 + I                                                           00001500
    16            35 WORK (K) = IN (I)                                                   00001600
    17               K = 1                                                               00001700
    18         C CONVERT LOOP                                                            00001800
    19            35 DO 100 I = K, 9                                                     00001900
    20               IF (HOLD (I) .EQ. BLANK) GO TO 200                                  00002000
    21               IF (HOLD (I) .NE. POINT) GO TO 40                                   00002100
    22               IF (PLG .GT. 0) GO TO 300                                           00002200
    23               PLG = 1                                                             00002300
    24               GO TO 200                                                           00002400
    25            40 IF (HOLD (I) .GT. UPLIM) GO TO 300                                  00002500
    26               IF (HOLD (I) .LT. LOLIM) GO TO 300                                  00002600
    27         C HERE IF VALID NUM                                                       00002700
    28            100 OUT (I) = (HOLD (I) - TEST) / 256                                  00002800
    29               GO TO 210                                                           00002900
    30         C ASSEMBLE ANSWER-SET K IN CASE OF DECIMAL AT 1                           00003000
    31            200 K = I + 1                                                          00003100
    32            201 I = I - 1                                                          00003200
    33            210 IF (PLG .GT. 1) GO TO 260                                          00003300
    34               IF (I) 290, 290, 220                                                00003400
    35            220 ANS = ANS + (OUT (I) * J)                                          00003500
    36               J = J * 10                                                          00003600
    37               GO TO 201                                                           00003700
    38         C END ANSWER LOOP                                                         00003800
    39         C TEST IF REAL OR FRACT                                                   00003900
    40            250 IF (PLG .LE. 0) GO TO 295                                          00004000
    41         C HERE IF HAVE DECODED REAL PORTION-NOW GET FRACTION                      00004100
    42               IF (K .GT. 9) GO TO 275                                             00004200
    43               K1 = K                                                              00004300
    44               PLG = 2                                                             00004400
    45               GO TO 35                                                            00004500
    46         C FRACTION LOOP                                                           00004600
    47            260 A = 1                                                              00004700
    48               DO 270 L = K1, I                                                    00004800
    49               IF ((I - L) .LT. 0) GO TO 275                                       00004900
    50               A = A * .1                                                          00005000
    51            270 FRACT = FRACT + (OUT (L) * A)                                      00005100
    52            275 DECMAL = ANS + FRACT                                               00005200
    53               CODE = 9                                                            00005300
    54               RETURN                                                              00005400
    55         C VALID INTEGER -RETURN                                                   00005500
    56            295 CODE = 0                                                           00005600
    57               RETURN                                                              00005700
    58         C ALPHA - SET ERROR                                                       00005800
    59            300 CODE = 10                                                          00005900
    60               RETURN                                                              00006000
    61               END                                                                 00006100
```

TABLE VI-9

08/14/75 INPUT LISTING -MSG-

LISTING - SUBROUTINE MSG

FORTRAN MODULE (LIST,NAMSUT)SUBNCSECT

CARD NO	CONTENTS	
1	C SUBROUTINE TO DECODE TEXT MSG FOR 'WHAT IF' (TPF) 13JUN75	00000100
2	C CODE- QUIT (1), DONE (2), HELP (3), STAT (4), SHOW (5), COPY (6),	00000200
3	C NEW (7), ALL (8), MOD (9), CODE NOT RECOGNIZED (99)	00000300
4	SUBROUTINE MSG (IN, CODE)	00000400
5	INTEGER*2 F(4,9), IN(5), CODE, QUIT(4), DONE(4), HELP(4), STAT(4)	00000500
6	INTEGER*2 SHOW (4), COPY (4), NEW (4), ALL (4), MOD (4)	00000600
7	EQUIVALENCE (QUIT,F(1,1)),(DONE,F(1,2)),(HELP,F(1,3)),(NEW,F(1,7))	00000700
8	EQUIVALENCE (SHOW,F(1,5)),(COPY,F(1,6)),(ALL,F(1,8)),(MOD,F(1,9))	00000800
9	EQUIVALENCE (STAT,F(1,4))	00000900
10	DATA QUIT /'Q','U','I','T'/, DONE /'D','O','N','E'/	00001000
11	DATA HELP /'H','E','L','P'/, SHOW /'S','H','O','W'/	00001100
12	DATA COPY /'C','O','P','Y'/, NEW /'N','E','W',' '/	00001200
13	DATA ALL /'A','L','L',' '/, MOD /'M','O','D',' '/	00001300
14	DATA STAT /'S','T','A','T'/	00001400
15	CODE = 99	00001500
16	DO 2 I = 1,9	00001600
17	DO 1 J = 1,4	00001700
18	IF (IN (J) .NE. F(J,I)) GO TO 2	00001800
19	1 CONTINUE	00001900
20	C HERE IF VALID COMPARE ON ALL 5 CHARACTERS	00002000
21	CODE = I	00002100
22	GO TO 99	00002200
23	2 CONTINUE	00002300
24	99 RETURN	00002400
25	END	00002500

TABLE VI-10

08/14/75 INPUT LISTING -SHOW-

LISTING - SUBROUTINE SHOW

FORTRAN MODULE (LIST,NAMSQ,SOURCSEQ)

```
CARD NO  ****  CONTENTS                                                    ****
     1   C DISPLAY CURRENT COURSE (TPFMOD), N IS CRS NO IN FILE 13JUN75     00000100
     2         SUBROUTINE SHOW (N)                                         00000200
     3         IMPLICIT INTEGER*2 (A)                                      00000300
     4         INTEGER*4 B1,B16,B35,B36                                    00000400
     5         REAL*4 B5,B11,B12,B13,B14,B18,B25,B31,B32                   00000500
     6         REAL*8 B2A,B3A,B3B                                          00000600
     7         COMMON /A/ B1(144),B2A(144),B3A(144),B4(144),B5(144),B6(144)  00000700
     8         COMMON /A/ B7(144),B8(144),B9(144),B10(144),B11(144),B12(144)  00000800
     9         COMMON /A/ B13(144),B14(144),B15(144),B16(144),B17(144),B18(144)  00000900
    10         COMMON /A/ B19(144),B20(144),B21(144),B22(144),B23(144),B24(144)  00001000
    11         COMMON /A/ B25(144),B26(144),B27(144),B28(144),B29(144),B30(144)  00001100
    12         COMMON /A/ B31(144),B32(144),B33(144),B34(144),B35(144),B36(144)  00001200
    13         COMMON /A/ B2B (144),B3B(144)                               00001300
    14         INTEGER*2 IN (5) /5*0/, BLANK /' '/, CODE                   00001400
    15         I = N                                                       00001500
    16         IF (I .LE. 0) GO TO 998                                     00001600
    17         IF (I .GT. 144) GO TO 998                                   00001700
    18       1 WRITE (6,2)                                                 00001800
    19       2 FORMAT (/' SHOW: ENTER ALL OR FIELD NAME')                  00001900
    20         READ (5,3) IN                                               00002000
    21       3 FORMAT (5A1)                                                00002100
    22         CALL MSG (IN, CODE)                                         00002200
    23         GO TO (999,999,10,10,10,10,10,100,10), CODE                 00002300
    24   C HERE IF NOT RECOGNIZED-TEST IF BLANK(A RETURN)                  00002400
    25         IF (IN (1) .EQ. BLANK) GO TO 100                            00002500
    26   C NOT RECOGNIZED MSG OR BLANK-TEST IF FIELD NAME                  00002600
    27         NN = 9999                                                   00002700
    28         CALL DECODE (NN,N,CODE,IN)                                  00002800
    29         IF (CODE .EQ. 0) GO TO 999                                  00002900
    30      10 WRITE (6,11) IN                                             00003000
    31      11 FORMAT (1X,5A1,' -INVALID- RETRY'/)                         00003100
    32         GO TO 1                                                     00003200
    33   C ****SHOW ALL ****                                               00003300
    34     100 I = N                                                       00003400
    35   C CALC STU BKLG- B4 CONV: B6 CAP: B7 BCAP: B15 BKLG WKS           00003500
    36         NBKL = 0                                                    00003600
    37         IF (B4(I) .LE. 0) GO TO 299                                 00003700
    38         ICAP = B6(I) - B7(I)                                        00003800
    39         IF (ICAP .LE. 0) GO TO 299                                  00003900
    40         CNVB = 50. /B4(I)                                           00004000
    41         NBKL = B15(I)/CNVB * ICAP                                   00004100
    42     299 CONTINUE                                                    00004200
    43   C **** PRINT IT ****                                              00004300
    44         WRITE (6,4001) B1(I),B3A(I),B3B(I)                          00004400
    45         WRITE (6,4002) B4(I),B17(I),B24(I),NBKL                     00004500
    46         WRITE (6,4003) B5(I),B10(I),B25(I),B15(I)                   00004600
    47         WRITE (6,4004) B36(I),B16(I),B2A(I),B2B(I),B35(I)           00004700
    48        *, B6(I),B19(I),B26(I)                                       00004800
    49         WRITE (6,4005) B7(I),B20(I),B27(I),B23(I)                   00004900
    50         WRITE (6,4006) B8(I),B21(I),B28(I),B30(I)                   00005000
    51         WRITE (6,4007) B31(I),B32(I),B34(I),B33(I),B9(I),B22(I),B29(I)  00005100
    52        *, B10(I)                                                    00005200
    53         RATEA = B11(I) * 100.                                       00005300
    54         RATEB = B12(I) * 100.                                       00005400
    55         RATEC = B13(I) * 100.                                       00005500
    56         WRITE (6,4008) RATEA, RATEB, RATEC                          00005600
    57    4001 FORMAT (//1X,A4, 5X, 2A8, 4X, '--SCHEDULE DATA--  FY  FY+1  FY+00005700
    58        *2  CRS BACKLOG')                                            00005800
    59    4002 FORMAT (' ', T31, 'ANNUAL CONVENINGS', I5, 2I7, 4X, 'B STU', I5)  00005900
    60    4003 FORMAT (' SCHL JNO  CATALOG NO  NEC   COURSE LENGTH DYS', F5.1,  00006000
    61        * 2F7.1, 4X, 'B WKS', I5)                                    00006100
    62    4004 FORMAT (' ', 2 (A4, 1X), A8, A2, 1X, A4, 4X, 'COURSE CAPACITY',  00006200
    63        * 3I7)                                                       00006300
    64    4005 FORMAT (' ', T31, 'BUPERS CAPACITY', 3I7, 3X, 'FY+1 WK', I4)  00006400
    65    4006 FORMAT (' P AOB  PFAIL  TYPE   PRTY   ANNUAL DEMAND', 2X, 3I7,  00006500
    66        * 3X, 'FY+2 WK', I4)                                         00006600
    67    4007 FORMAT (' ', F5.1, F6.1, '%', 3X,A2,5X,A2,5X, 'BUPERS DEMAND',  00006700
    68        * 2X, 3I7, 3X, 'OFFSET', I5)                                 00006800
    69    4008 FORMAT (/' RATES: FAIL',F5.1,'%  NADIS',F5.1,'%  NOSHOW',F5.1,'%00006900
    70        *'///)                                                       00007000
    71         GO TO 999                                                   00007100
    72     998 WRITE (6,98) I                                             00007200
    73      98 FORMAT (/1X,I3,' COURSE NOT IN PROCESS- SHOW SKIPPED'/)     00007300
    74     999 RETURN                                                      00007400
    75         END                                                         00007500
    76         BLOCK DATA                                                  00007600
    77         IMPLICIT INTEGER*2 (A)                                      00007700
    78         INTEGER*4 B1,B16,B35,B36                                    00007800
    79         REAL*4 B5,B11,B12,B13,B14,B18,B25,B31,B32                   00007900
    80         REAL*8 B2A,B3A,B3B                                          00008000
    81         COMMON /A/ B1(144),B2A(144),B3A(144),B4(144),B5(144),B6(144)  00008100
    82         COMMON /A/ B7(144),B8(144),B9(144),B10(144),B11(144),B12(144)  00008200
    83         COMMON /A/ B13(144),B14(144),B15(144),B16(144),B17(144),B18(144)  00008300
    84         COMMON /A/ B19(144),B20(144),B21(144),B22(144),B23(144),B24(144)  00008400
    85         COMMON /A/ B25(144),B26(144),B27(144),B28(144),B29(144),B30(144)  00008500
    86         COMMON /A/ B31(144),B32(144),B33(144),B34(144),B35(144),B36(144)  00008600
    87         COMMON /A/ B2B (144),B3B(144)                               00008700
    88         DATA B1,B16,B35,B36/5760* '/,B2A,B3A,B3B/4320* '/,B2B/' */  00008800
    89         DATA B5,B11,B12,B13,B14,B18,B25,B31,B32 /1296*0./           00008900
    90         DATA B4,B6,B7,B8,B9,B10,B15,B17,B19,B20,B21,B22,B23,B24,B26,  00009000
    91        1 B27, B28, B29, B30,  B34 /2880 * 0/, B33 /' '/             00009100
    92         END                                                         00009200
```

TABLE VI-11

SUBROUTINE COMND LISTING

```
COMND    START
         STM     14,12,12(13)
         BALR    12,Ø
         USING   *,12
         LM      3,4,Ø(1)
         LH      5,Ø(3)
         C       5,=F'17'
         MVC     Ø(2,4),=X'ØØØ1'
         BC      2,END
         LA      7,TAB1
         AR      5,7
         MVC     MSG+1(1),Ø(5)
         LA      L,PLIST
         SVC     X'CA'
         DC      AL4(ERR)
         MVC     Ø(2,4),=X'ØØØØ'
         B       END
ERR      MVC     Ø(2,4),=X'ØØØ2'
END      MVI     12(13),X'FF'
         LM      14,12,12(13)
         SR      15,15
         BR      14
         DS      ØF
PLIST    DC      AL8'TYPE'
         DC      X'81'
         DC      AL3(MSG)
         DC      C'B'
         DC      AL3(2)
MSG      DC      X'7LØØ'
TAB1     DC      X'718848C9995939B878F9FFØ3ØAØCØFFAF3Ø5'
         END
```

TABLE VI-12

FLOW - EXECUTIVE TPF

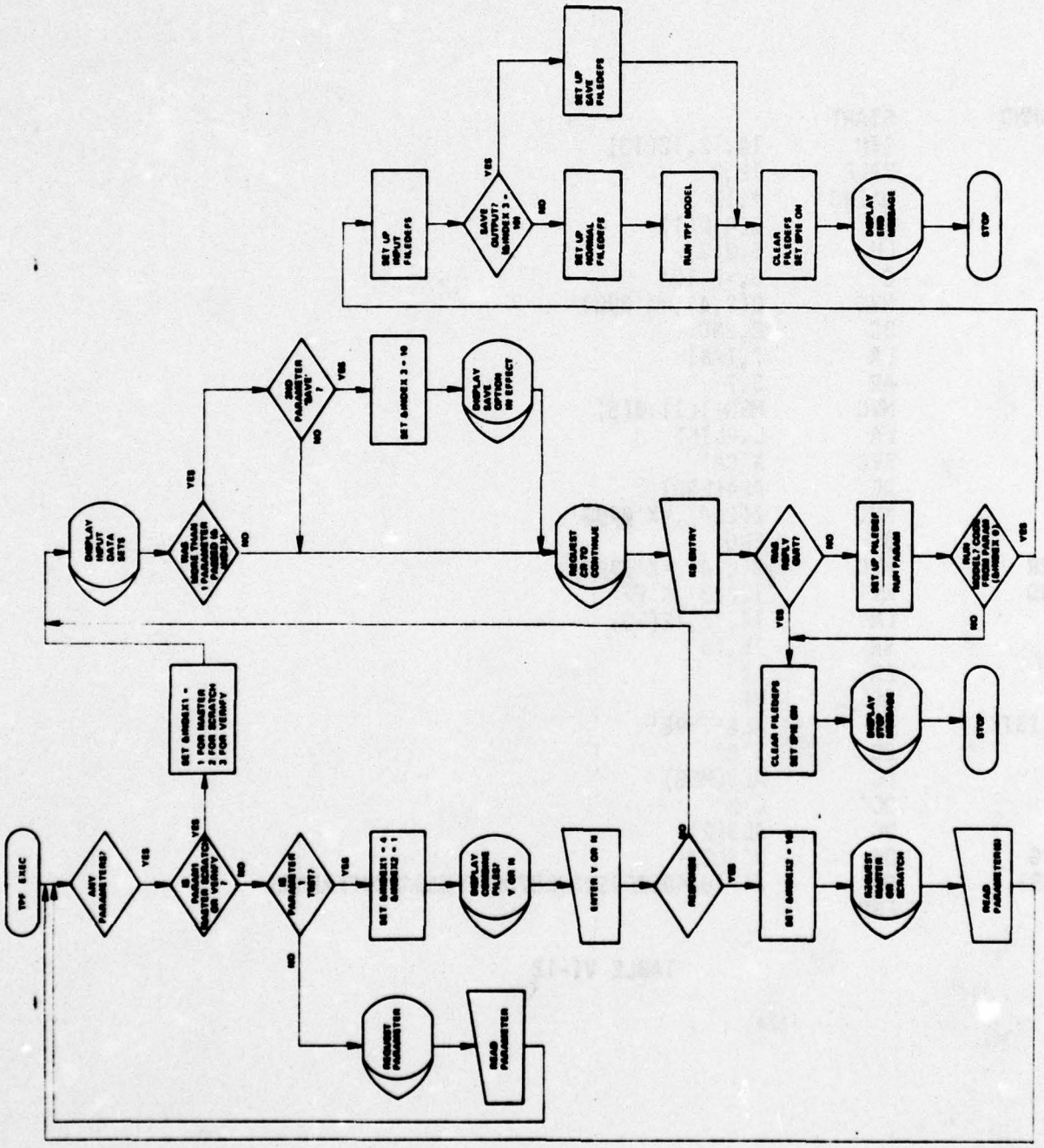


FIGURE VI-6

LISTING - EXECUTIVE TPF

TPF EXEC

```
&TYPE OFF
-TESTIT &IF &INDEX EQ Ø &GOTO -NONE
&IF &1 EQ QUIT &GOTO -QUIT
&INDEX1 = 1
&IF &1 EQ MASTER &GOTO -GOTIT
&INDEX1 = 2
&IF &1 EQ SCRATCH &GOTO -GOTIT
&INDEX1 = 3
&IF &1 EQ VERIFY &GOTO -GOTIT
&INDEX1 = 4
&IF &1 EQ TEST &GOTO -TEST
&IF &1 EQ QUIT &EXIT
&IF &1 EQ COPY &PRINT TPF MODEL- COPY INVALID- ENTER QUIT TO EXIT
&IF &1 EQ MOD &PRINT TPF MODEL- MOD INVALID- ENTER QUIT TO EXIT
-NONE &PRINT REPLY MASTER, SCRATCH, OR TEST (INPUT DATA)
&READ ARGS
&GOTO -TESTIT
-TEST &PRINT COMBINE TEST FILE WITH MASTER OR SCRATCH? (Y OR N)
&INDEX2 = 1
&READ ARGS
&IF &1 NE Y &GOTO -GOTIT
&INDEX2 = 1Ø
&PRINT REPLY MASTER OR SCRATCH
&READ ARGS
&GOTO -TESTIT
-GOTIT &PRINT
&PRINT INPUT THIS RUN:
&IF &INDEX1 EQ 1 &PRINT MASTER
&IF &INDEX1 EQ 2 &PRINT SCRATCH
&IF &INDEX1 EQ 3 &PRINT VERIFY
&IF &INDEX1 EQ 4 &PRINT TEST
&IF &INDEX2 EQ 1Ø &PRINT COMBINED WITH TEST
&IF &INDEX LT 2 &GOTO -NOSAVE
&IF &2 NE SAVE &GOTO -NOSAVE
&INDEX3 = 1Ø
&PRINT SAVE OPTION IN EFFECT THIS RUN
-NOSAVE &PRINT ENTER CARRIAGE RETURN TO CONTINUE
&READ ARGS
&IF &INDEX EQ Ø &GOTO -PARM
&IF &1 EQ QUIT &GOTO -QUIT
-PARM FILEDEF * CLEAR
FILEDEF PARM DSK TPF PARM
PARAM
&IF &INDEXØ GE 2 &GOTO -QUIT
FILEDEF * CLEAR
FILEDEF 5 DSK TPF PARM
FILEDEF 6 DUM
FILEDEF 8 DUM
&IF &INDEX1 EQ 1 FILEDEF 1Ø DSK TPF MASTER
&IF &INDEX1 EQ 2 FILEDEF 1Ø DSK TPF SCRATCH
&IF &INDEX1 EQ 3 FILEDEF 1Ø DSK TPF VFY
&IF &INDEX1 EQ 4 FILEDEF 1Ø DSK TST DATA
&IF &INDEX2 EQ 1Ø FILEDEF 1Ø DSK TST DATA CONCAT
&IF &INDEX3 EQ 1Ø &GOTO -SAVEOUT
FILEDEF 9 DSK OUT4 FILE LR 81
FILEDEF 11 DSK OUT3 FILE LR 81
&GOTO -MODEL
-SAVEOUT FILEDEF 9 DSK OUT2 FILE LR 81
FILEDEF 11 DSK OUT1 FILE LR 81
-MODEL RUN KEND (BRIEF)
FILEDEF * CLEAR
SET SPIE ON
&PRINT END TPF EXECUTION
&PRINT ENTER: CTRRUN FOR CENTER SUMMARY; CRSRUN FOR COURSE OUTPUTS
&EXIT
-QUIT FILEDEF * CLEAR
SET SPIE ON
&PRINT MODEL NOT RUN AT OPERATOR REQUEST
&EXIT
```

TABLE VI-13

LISTING - PROGRAM PARAM

```
 PARAM: PROC OPTIONS (MAIN) ;
 /* TPF PARAMETER CARD MOD PROGRAM 06FEB76 */
 DCL (PCARD,DATA) CHAR (80), REQ CHAR (10), LINE(8) CHAR (80) VARYING,
     FIELDS (24) CHAR (5) INITIAL
    ('FAIL','NOSHO','DMD','TREND','CONV','LEN','CAP','SETB',
     'CDP','SCH','PRINT','WKS','TRAC','PRE','RELS','END','STAT',
     'CRS','SCHL','CTR','INPUT','FYR','LYR','AOB'),
     LEN (24) FIXED BIN (8,0) INITIAL
     ((10)4,(10)1,2,(3)4),
     COL(24) FIXED BIN (8,0) INITIAL (16,21,26,31,36,41,
     46,56,61,66,71,72,73,74,75,76,77,78,79,80,1,3,7,11),
     (K,LP,I,P1,P2) FIXED BIN (8,0) ;
DCL FIELD6 CHAR (6), FIELD4 CHAR (4) DEF PCARD POS (52);
 DCL COMND EXTERNAL ENTRY;
 DCL IHESARC ENTRY (FIXED BIN (31,0));
 DCL (II,JJ) FIXED BIN (15,0);
 OPEN FILE (PARM) RECORD UPDATE ;
 READ FILE (PARM) INTO (PCARD) ;
II = 6; CALL COMND (II,JJ);
 INTRO:
 DISPLAY ('TPF MODEL - WANT INSTRUCTIONS (Y OR N)') REPLY (DATA);
 IF DATA = 'N' THEN GO TO PRERUN ;
 HELP: DISPLAY (' ');
 DISPLAY ('ENTER: FNAME XXXX   (SET FNAME TO VALUE)');
 DISPLAY ('        FNAME DEACT (DEACT FNAME)');
 DISPLAY ('        SHOW        (SHOW RUN COND)');
 DISPLAY ('        RUN         (RUN MODEL)');
 DISPLAY (' ');
 DISPLAY ('FIELDS:  FYR    (NNNN)  LYR    (NNNN)  AOB    (NNNN)  CDP    (AA
AA)  SCH    (AAAA)');
 DISPLAY ('         FAIL   (NNNN)  NOSHO  (NNNN)  DMD    (NNNN)  TREND  (NN
NN)  CONV   (NNNN)');
 DISPLAY ('         LEN    (NNNN)  CAP    (NNNN)  SETB   (NNNN)');
 DISPLAY (' ');
 DISPLAY ('REPORTS: CRS    (N)     SCHL   (N)     CTR (N)        PRINT (N)
');
PRERUN: FLAG = 2; GO TO TUBE_IT;
 COME_BACK: FLAG = 0; DISPLAY (' ');
 DISPLAY ('ENTER RUN, QUIT, OR FIELD TO BE CHANGED');
 PARMS: DISPLAY (' ');
 DISPLAY ('ENTER REQUEST:') REPLY (DATA) ;
 IF DATA = 'RUN' THEN GO TO FILE_IT ;
 IF DATA = 'FLAG' THEN GO TO FLG;
 IF DATA = 'HELP' THEN GO TO HELP;
 IF DATA = 'SHOW' THEN GO TO TUBE_IT ;
 IF DATA = 'QUIT' THEN GO TO QUIT;
 P1 = INDEX (DATA,' ') ;
 P2 = INDEX (SUBSTR (DATA,P1+1),' ') ;
 REQ = SUBSTR (DATA,1,P1-1) ;
 DO I = 1 TO 24 ;
 IF REQ = FIELDS (I) THEN GO TO GOT_IT ;
 END;
 DISPLAY ('FIELDNAME NOT RECOGNIZED') ;
 GO TO PARMS ;
 GOT_IT: IF SUBSTR (DATA,P1+1,6) = 'DEACT ' THEN GO TO DEAC ;
```

LISTING - PROGRAM PARAM (Continued)

```
IF SUBSTR (DATA,1,P1-1) = 'SCH' THEN
IF P2 < 5 THEN P2 = 5;
IF P2-1 ¬= LEN(I) THEN GO TO ERR2 ;
IF I <= 10 THEN GO TO FLAG_AND_VALU :
SUBSTR (PCARD,COL(I),LEN(I)) = SUBSTR (DATA,P1+1,LEN(I)) ;
GO TO PARMS ;
TUBE_IT: DISPLAY (' '); DISPLAY ('PRESENT RUN CONDITIONS');
DO K = 1 TO 8 ;
LINE(K) = '' ;
END;
LP = 0 ;
DO K = 1 TO 7 ;
IF SUBSTR (PCARD,11+5*K,1) ¬= '0' THEN DO ;
LINE (LP/3 + 1) = LINE (LP/3 + 1) || FIELDS(K) ||
 ' = ' || SUBSTR (PCARD,12+5*K,4) || '     ' ;
LP = LP + 1 ;
END ; END;
DO K = 8 TO 10 ;
IF SUBSTR (PCARD,16+5*K,1) ¬= '0' THEN DO ;
LINE (LP/3 + 1) = LINE (LP/3 + 1) || FIELDS(K) ||
 ' = ' || SUBSTR (PCARD,17+5*K,4) || '     ' ;
LP = LP + 1 ;
END ; END;
LP = ((LP+3)/3) * 3;
LINE (LP / 3 + 1) = 'REPORTS: ';
FLAG1 = 0;
DO K = 11,18,19,20;
IF SUBSTR (PCARD,COL (K),1) ¬='0' THEN DO;
FLAG1 = 10;
LINE (LP/3 + 1) = LINE (LP/3 + 1) || ' ' || FIELDS (K);
END; END;
IF FLAG1 ¬= 10 THEN
LINE (LP/3 + 1) = LINE (LP/3 + 1) || '** NONE **';
LP = LP + 3;
LINE (LP/3 + 1) = 'INPUT = ' || SUBSTR (PCARD,1,2) || '     FYR   = '
 || SUBSTR (PCARD,3,4) || '     LYR   = ' || SUBSTR (PCARD,7,4)
 || '     AOB   = ' || SUBSTR (PCARD,11,4) ;
K = (LP/3) + 1 ;
DO I = 1 TO K ;
DISPLAY (LINE(I)) ;
END ;
IF FLAG = 2 THEN GO TO COME_BACK;
GO TO PARMS ;
FLAG_AND_VALU:
SUBSTR (PCARD,COL(I),LEN(I)+1) =
'1' || SUBSTR (DATA,P1+1,LEN(I)) ;
GO TO PARMS ;
DEAC: IF I > 20 THEN GO TO ERR1 ;
SUBSTR (PCARD,COL(I),1) = '0' ;
```

TABLE VI-14 (Continued)

LISTING - PROGRAM PARAM (Continued)

```
 GO TO PARMS ;
 ERR1: DISPLAY ('INVALID REQUEST') ;
 GO TO PARMS ;
 ERR2: DISPLAY ('INVALID DATA FORMAT') ;
 GO TO PARMS ;
FLG: DISPLAY ('FLAGS - WKS TRAC PRE RELS END STAT YEAR PRINT INIT');
 GO TO PARMS;
 QUIT: II = 8; CALL IHESARC (II);
 FILE_II: FIELD6 = DATE; FIELD4 = SUBSTR (FIELD6,3,4);
 REWRITE FILE (PARM) FROM (PCARD) ;
 CLOSE FILE (PARM) ;
 END PARAM ;
```

TABLE VI-14 (Continued)

DISPLAY SUPPORT PROGRAM

The objective of the display support program is to allow the user to effectively interface with the large amounts of data that the TPF model may produce. This support package is general purpose in nature, and is useful for displaying the contents of other data sets as well.

EXECUTIVE DISP. Executive DISP is used to control the execution of program DISPLAY. This short executive accepts a parameter passed from the user or calling executive and uses this parameter to define the data set to be displayed. FORTRAN data set 7 is used as the output data set and is termed CONO by NCSS convention to label it the console or terminal device. This output data set is used rather than the conventional data set 6 so that FORTRAN I/O error messages will not be displayed. SET SPIE ON is specified in this executive as the FORTRAN I/O error checking is disabled by internal subroutines, and this condition must be reset. This executive is usually called by executives CTRRUN, CRSRUN, CENTER, or COURSE. These executives are three-line executives that merely pass the proper file name to the DISP exec. CTRRUN and CRSRUN are used to display center and course data respectively for TPF runs where the SAVE option is not specified, while CENTER and COURSE are used to display those data if SAVE was specified.

PROGRAM DISPLAY. Program DISPLAY is a FORTRAN program used to display the contents of TPF output data sets. The program ascertains the format of the output data, determines page size, and presents these data along with options to the user. A standard TPF output data set is in the form:

CTL CHAR(1)	DATA (79)
1	First line of page
Blank	Data (variable number of lines)
/	Last line of data
Blank	Null line
1	First line of next page

Program DISPLAY first searches the first column of each record for a "/", to determine if a TPF data set is to be displayed, and if so, to ascertain the size of each page. Next the program skips through the file to determine the number of pages in the file. Finally, if the file contains 25 lines per page, it is assumed to be a course file, and the CDP options are enacted. If the file is not a TPF file, the program defaults to 20 lines per page.

The user has the option of specifying the page number to be displayed, or entering a carriage return to display the next (or first) page. The user may also enter LOOK, followed by a blank and a page number to display only the first line of a page. Finally, if the file is a course file, the user may specify the CDP number of the course to be displayed.

Direct access I/O is used by this program in locating and displaying data so as to allow the user to browse back and forth through the data. All terminal responses are read in the "A" format as in program TPFMOD1 to allow the user to intermix alpha and numeric data and to avoid left or right justification. Subroutine CONVRT, detailed in the section describing TPFMOD1 is used to convert these input data to a machine processable form. Subroutine COMND, also described in that section is used to clear the display screen prior to a page

display. In-line subroutine RANGE is used to intercept FORTRAN errors should the user specify a page outside the boundaries of the input data set, while in-line subroutine ALPHA intercepts any I/O conversion errors in order to provide smooth operation to the user.

LISTING - DISPLAY EXECUTIVES

COURSE EXEC

```
&TYPE OFF
EXEC DISP OUT2
&EXIT
```

CENTER EXEC

```
&TYPE OFF
EXEC DISP OUT1
&EXIT
```

CRSRUN EXEC

```
&TYPE OFF
EXEC DISP OUT4
&EXIT
```

CTRRUN EXEC

```
&TYPE OFF
EXEC DISP OUT3
&EXIT
```

DISP EXEC

```
&TYPE OFF
FILEDEF * CLEAR
FILEDEF 3 DSK &1 FILE
FILEDEF 7 CONO
DISPLAY
SET SPIE ON
FILEDEF * CLEAR
&EXIT
```

08/14/75 INPUT LISTING -DISPLAY-

LISTING - PROGRAM DISPLAY

FORTRAN MODULE (LIST,NAMSQ,SOURLSEQ)

```
CARD NO    ****                    CONTENTS                    ****
     1     C TPF SUPPORT - UPDATE 22MAY75                                00000100
     2     C DISPLAY- DISPLAYS OUTPUT FILE FOR 3 MODES (AUTO SELECT OF STD/NON-STD)00000200
     3     C STANDARD FILE CONTAINS 1 IN COL 1 AT START OF PAGE, / AFTER LAST LINE 00000300
     4     C 1. COURSE FILE (25 LINES/PAGE) SELECT BY PAGE OR CDP.       00000400
     5     C 2. SCHOOL/CENTER FILE - SELECT BY PAGE NO.                  00000500
     6     C 3. NON STANDARD FILE - SELECT BY PAGE NO; DEFAULT TO 20 LINES/PAGE  00000600
     7           INTEGER*2 IN(5), IN2(5), CODE, TEST(4,5), SIZE,NSTART,NNN,ENDIT 00000700
     8           DATA TEST/'0','U','I','T','D','O','N','E','L','O','O','K',  00000800
     9          1 'P','A','G','E','C','D','P',' '/                         00000900
    10           INTEGER*2 BLANK /' '/                                      00001000
    11           REAL*4 CDP, DONE, PAGE, QUIT, SELECT                       00001100
    12           REAL*8 LINE (10)                                           00001200
    13           INTEGER*4 NUM                                              00001300
    14           COMMON NGA,NGR                                             00001400
    15           EXTERNAL RANGE                                             00001500
    16           EXTERNAL ALPHA                                             00001600
    17           DEFINE FILE 3 (2500,81,E,J3)                               00001700
    18           DATA SLASH/'/'/,CDP/'CDP '/,DONE/'DONE'/,PAGE/'PAGE'/,QUIT/'QUIT'/00001800
    19           CALL ERRSET (232,100,-1,1,RANGE)                           00001900
    20           CALL ERRSET (215,100,-1,1,ALPHA)                           00002000
    21           NGA = 0                                                    00002100
    22           NGR = 0                                                    00002200
    23           NSTART = - 1                                               00002300
    24           LOOKFL = 0                                                 00002400
    25           NSET = 2                                                   00002500
    26           NNN = 6                                                    00002600
    27           CALL COMND (NNN,J1)                                        00002700
    28     C LOAD PAGE SIZE TO CALC POSITION                                00002800
    29           DO 20 I = 1, 26                                            00002900
    30           READ (3'I,15,ERR=999) CTLCAR                               00003000
    31        15 FORMAT (A1,80X)                                            00003100
    32           IF (CTLCAR .EQ. SLASH) GO TO 30                            00003200
    33        20 CONTINUE                                                   00003300
    34     C HERE IF NO SLASH IN 1 - INVALID FILE                           00003400
    35           WRITE (7,25)                                               00003500
    36        25 FORMAT (' NON STANDARD FILE TYPE - DEFAULT 20 LINES PER PAGE') 00003600
    37           SIZE = 20                                                  00003700
    38           NSET = 0                                                   00003800
    39           GO TO 26                                                   00003900
    40     C                                                                00004000
    41        30 SIZE = I + 1                                               00004100
    42           WRITE (7,35) SIZE                                          00004200
    43        35 FORMAT (' STANDARD FILE - ',I2,' LINES')                   00004300
    44     C DETERMINE NUMBER OF PAGES IN FILE                              00004400
    45        26 DO 10026 I = 1, 100                                        00004500
    46           N = I * SIZE + 1                                           00004600
    47           READ (3'N,15,ERR=999) CTLCAR                               00004700
    48           IF (NGR) 10026, 10026, 10027                               00004800
    49     10026 CONTINUE                                                   00004900
    50     10027 NGR = 0                                                    00005000
    51           WRITE (7,10028) I                                          00005100
    52     10028 FORMAT (1X, I3, ' PAGES IN FILE')                          00005200
    53           IF (SIZE .NE. 25) GO TO 27                                 00005300
    54           IF (I .EQ. 1) GO TO 27                                     00005400
    55           WRITE (7, 40)                                              00005500
    56        40 FORMAT (/' +COURSE FILE - SELECT BY ''CDP'' OR ''PAGE'' ') 00005600
    57           GO TO 241                                                  00005700
    58        27 WRITE (7,36)                                               00005800
    59        36 FORMAT (/' +ENTER PAGE NUMBER')                            00005900
    60       241 READ (5,5) IN,IN2                                          00006000
    61         5 FORMAT (10A1)                                              00006100
    62           INFLG = 0                                                  00006200
    63     C TEST IF IN IS NUMBER OR CODE                                   00006300
    64       242 CALL CONVRT (IN,NUM,CODE,DUM)                              00006400
    65           IF (CODE .LE. 5) GO TO 666                                 00006500
    66     C HAS ALPHAS- SEE IF VALID CODE                                  00006600
    67           IF (INFLG .GT. 0) GO TO 247                                00006700
    68           INFLG = 10                                                 00006800
    69           DO 244 I = 1,5                                             00006900
    70           DO 243 J = 1,4                                             00007000
    71           IF (IN (J) .NE. TEST(J,I)) GO TO 244                       00007100
    72       243 CONTINUE                                                   00007200
    73     C HERE IF RECOGNIZED CODE                                        00007300
    74           GO TO (999,999,245,246,800),I                              00007400
    75           GO TO 247                                                  00007500
    76       244 CONTINUE                                                   00007600
    77       247 WRITE (6,6) IN                                             00007700
    78         6 FORMAT (1X,5A1,' +INVALID- RETRY'/)                        00007800
    79           GO TO 241                                                  00007900
    80     C HERE IF LOOK- NEXT LINE IF PAGE                                00008000
    81       245 LOOKFL = 10                                                00008100
    82     C NOW TEST IF IN2 IS BLANK OR HAS NUM                            00008200
    83       246 DO 248 I = 1,5                                             00008300
    84           IF (IN2(I) .NE. BLANK) GO TO 249                           00008400
    85       248 CONTINUE                                                   00008500
    86     C HERE IF BLANK- GO GET NUM                                      00008600
    87           GO TO 27                                                   00008700
    88     C HAVE SECOND ENTRY- LOAD INTO IN & DECODE                       00008800
    89       249 DO 250 I = 1,5                                             00008900
    90       250 IN (I) = IN2 (I)                                           00009000
    91           INFLG = 10                                                 00009100
    92           GO TO 242                                                  00009200
    93       666 IF (NUM .NE. 0) GO TO 1666                                 00009300
    94           NSTART = NSTART + SIZE                                     00009400
    95           NUM = NSTART / SIZE + 1                                    00009500
    96     C CALC NSTART POSITION OF PAGE                                   00009600
    97      1666 NSTART = (NUM - 1) * SIZE + 1                              00009700
    98       667 ENDIT = NSTART + (SIZE - NSET)                             00009800
    99           IF (LOOKFL .GT. 1) ENDIT = NSTART                          00009900
   100     C CLEAR PAGE THEN OUTPUT                                         00010000
   101           CALL COMND (NNN,J1)                                        00010100
   102           IF (LOOKFL .EQ. 0) WRITE (7,103) NUM                       00010200
   103       103 FORMAT (' PAGE', I4)                                       00010300
   104           LOOKFL = 0                                                 00010400
   105           DO 110 I = NSTART,ENDIT                                    00010500
   106           READ (3'I,115,ERR=900) CTLCAR,LINE                         00010600
   107       115 FORMAT (A1,9A8,A7)                                         00010700
   108           IF (NGR) 120,120,900                                       00010800
   109       120 IF (CTLCAR .EQ. SLASH) GO TO 140                           00010900
```

TAEG REPORT NO. 36

08/14/75 INPUT LISTING -DISPLAY-

CARD NO	CONTENTS	----
110	110 WRITE (7,116) LINE	00011000
111	116 FORMAT (1X,9A8,A7)	00011100
112	C	00011200
113	140 WRITE (7,39) NUM	00011300
114	GO TO 241	00011400
115	39 FORMAT (' PAGE', I4,' +PAUSING- ENTER NEXT PAGE NO.')	00011500
116	C ALPHA ERROR	00011600
117	200 NGA = 0	00011700
118	WRITE (6,6) IN	00011800
119	GO TO 241	00011900
120	C RANGE ERROR	00012000
121	300 NGR = 0	00012100
122	WRITE (7,310) NUM	00012200
123	310 FORMAT (1X,I4,1X,'OUT OF RANGE - RETRY')	00012300
124	GO TO 241	00012400
125	C	00012500
126	C COP SELECT ROUTINE	00012600
127	800 WRITE (7,802)	00012700
128	802 FORMAT (/' +ENTER COP')	00012800
129	803 READ (5,42,ERR=800) SELECT, IN2	00012900
130	42 FORMAT (A4,1X,5A1)	00013000
131	IF (SELECT .EQ. QUIT) GO TO 999	00013100
132	IF (SELECT .EQ. DONE) GO TO 999	00013200
133	IF (SELECT .EQ. PAGE) GO TO 246	00013300
134	C SEARCH IF COP IN FILE	00013400
135	DO 816 I = 1, 100	00013500
136	N = ((I - 1) * SIZE) + 1	00013600
137	READ (3'N,806,ERR=850) COPA	00013700
138	806 FORMAT (1X, A4)	00013800
139	IF (NGR)810, 810, 850	00013900
140	810 IF (COPA .EQ. SELECT) GO TO 820	00014000
141	816 CONTINUE	00014100
142	C	00014200
143	C HAVE COP IN FILE - PRINT PAGE	00014300
144	820 CALL COMND (NNN,JI)	00014400
145	NUM = 1	00014500
146	ENDIT = N + 25	00014600
147	DO 825 I = N, ENDIT	00014700
148	READ (3'I,115,ERR=850) CTLCAR,LINE	00014800
149	IF (NGR) 822, 822, 850	00014900
150	822 IF (CTLCAR .EQ. SLASH) GO TO 830	00015000
151	825 WRITE (7,116) LINE	00015100
152	C END OF PAGE	00015200
153	830 WRITE (7,832) NUM	00015300
154	832 FORMAT (' PAGE',I4,' +PAUSING- ENTER COP, PAGE, OR QUIT')	00015400
155	GO TO 803	00015500
156	C HERE IF BAD READ / NO SELECT / OUT OF RANGE	00015600
157	850 NGR = 0	00015700
158	WRITE (7, 852) SELECT	00015800
159	852 FORMAT (1X, A4, ' +NOT IN FILE OR INVALID - RETRY')	00015900
160	GO TO 803	00016000
161	C WRAPUP (END)	00016100
162	999 WRITE (7,998)	00016200
163	998 FORMAT (' +DISPLAY TPF ENDED')	00016300
164	STOP	00016400
165	END	00016500
166	SUBROUTINE RANGE (KK,LL)	00016600
167	COMMON NGA,NGR	00016700
168	NGR = 10	00016800
169	KK = 0	00016900
170	LL = 232	00017000
171	RETURN	00017100
172	END	00017200
173	SUBROUTINE ALPHA (KK,LL)	00017300
174	COMMON NGA,NGR	00017400
175	NGA = 10	00017500
176	KK = 0	00017600
177	LL = 215	00017700
178	RETURN	00017800
179	END	00017900

TABLE VI-16 (Continued)

TPF LOGIC DESIGN

The design of the TPF model is such that it might be termed a front-end loaded model. That is, all schedule and demand conditions for a course are loaded at the beginning of the year, and the model then projects the resultant conditions for the remainder of the year. Likewise, the basic model projects, rather than predicts. The model itself is a true simulation, or mathematical representation of the mechanics of the overall scheduling function for the center. All major functions that affect student flow are modeled to a detail similar to the actual operations at the center, which include quota control, BUPERS detailing, scheduling, failures, and setbacks. The model outputs, however, can be predictive if the inputs themselves are predictive. To use the model for decision-making purposes, the user must input his predictions of demands, schedule changes, new courses, etc., and the model then projects the annual throughputs, failures, AOB's, and such, based on these predictions. Likewise, a simulation model is not an optimization model; the large number of variable input parameters precludes this. The user can manipulate the model inputs to achieve optimal results.

PROGRAM LOGIC DESIGN. The TPF model consists of a main control program and seven subroutines, all coded in the FORTRAN language. This structure was chosen for several reasons. All input data enter through the main control program, which edits the input data and builds a matrix for communication between the subroutines. Thus, if input methods should be changed, such as entering backlog as students rather than the present method of entering weeks, only one routine would be required. As another example, the entire scheduling routine is contained in subroutine CRSRLS. This subroutine represents the methods and procedures currently in use at the center. Should it be desirable to represent other centers with different methods and procedures, a new subroutine to represent that center could be written, and the appropriate subroutine could be linked to the model at execute time. In summary, the model has been written in a modular form so as to allow adaptability and ease of modification without extensive rewrite.

Certain programming conventions are followed throughout the design of the TPF model and its subroutines. All input data are read in at the beginning of the operation, with proper editing and error checking provided at that time. Parameter information is read in and various flags are set to control subsequent execution. In both cases, assumptions are made for missing or inaccurate data and processing continues with appropriate error messages. Finally, all communication throughout the model is through COMMON storage, with minimal use made of subroutine parameter passing.

TPF MODEL MAIN. The TPF model is compiled and stored for execution under the name KEND. This is to allow executives to use the name TPF. The basic control routine in this FORTRAN program is called MAIN. This module is the primary interface of the TPF model with the rest of the TPF system All input data and parameters are entered and edited by this section, and all control of the TPF model execution is set up by MAIN. MAIN also controls all cycling for the various loops necessary for the TPF simulation.

Parameter Options. The first operation in MAIN is to read in the control parameter card. This method of control was chosen to provide for card batch

operation as well as on-line terminal execution. The format of the parameter card is shown in Figure VI-3C. Four elements of the parameter card are key to execution; input, first year, last year, and AOB constant, as they set up control for execution. Input specifies whether the data are to be read from cards (05), or from an on-line data set (10), with the default value being 10. First year and last year are used to control the years of the projection (default 1 year, maximum 3 years) and the years entered are printed as titles on output reports and control data base access. AOB constant is used to denote the method and value to be used for average-on-board calculations. A value of 300 or greater will cause weekend days to be counted in AOB calculations, while a value of less than 300 will alter the model so that only working days are counted.

The next 8 parameters on the parameter card control across-the-board modification. The first character in each field is a flag (0 for off, 1 for active) and the remaining four characters are a multiplier. If the flag is 1 or greater, the multiplier is passed to other routines for data overrides. The next two fields allow the user to select only a course CDP number or school for execution, without the need to alter the data base. The date field contains the month and day of the run, as set by program PARAM.

The last 10 fields on this card contain flags used to control various options of the TPF model. If the INIT flag is set, the model will perform the pre-run initilization described later in this section. The output data provided is controlled by flags COURSE($R1), SCHOOL($R2), and CENTER($3). A 1 in any of these fields will produce the specified reports, while all zeros in these fields will cause the model to cycle with no output data generated. The rest of the flags control trace routines used for detailed analysis of operation of each of the TPF model subroutines. Use of these trace flags is not recommended for normal operation due to the volume produced. They may be set, however, using program PARAM and their names may be displayed by the program by requesting FLAGS.

Data Entry and Editing. The next major section of program MAIN consists of the logic to read data into storage, provide reasonableness checks, and to perform data overrides. All data are read into main storage to reduce disk I/O operations as the system runs under virtual storage. Each data element is tested for reasonableness and the value is clamped if necessary, allowing the model to execute using a data base with known errors. The various overrides are performed next for those fields where the override flag is set. Logic is included to assure that the overrides produce meaningful results, i.e., course convenings clamped to integer values, rather than allowing a value such as 2.87 convenings. Following these operations, MAIN passes control to the cycling portion of the model.

Model Control and Cycling. Except for one routine, the test of the main program logic is straightforward, using standard FORTRAN coding techniques. Three major cycling paths exist in the model; the year loop, the quarter loop, and the week loop. One of these, the quarterly loop, can undergo alteration to initialize the model. The TPF model does not snapshot existing conditions, but

rather it views conditions over a period of time. It considers the schedule and classes as they are today, and projects for one year. One important factor must be recognized. Today there may be 70 courses in session that were started in a preceding period and which, because of their length, terminate in the period being projected. To represent this, the quarter loop can reconfigure and, for the first quarter loop, the model operates as though it were the quarter prior to model start. No totals are accumulated for the run; the only effect is to release courses so that they may be active at model start. Subroutine PRERUN is used to accomplish this, and this initialization will be discussed in further detail in that segment.

The lowest order cycling loop in the model is the week loop. An understanding of how this loop operates will probably lead to an understanding of the entire model. This loop first scans each course loaded in the model, one at a time, to ascertain if a course should convene that week. It does this by testing the convening frequency of the course against a matrix loaded in core, for convening frequencies of 1 to 50 convenings per year. Before this comparison is made, the matrix is rationalized for offset, change week, and multiple convenings. This routine is described immediately following this segment, but for now, all that is necessary to know is that if this matrix returns a code that the course should start, subroutine PRERUN will be called and the course started. When a course is started, it is placed in an active matrix, which is described in the subroutine PRERUN segment. Upon completion of the test for possible convening of all courses, the MAIN program then tests all active courses to see if any end in that week. As the model cycles on a weekly basis, many courses will begin and end in the same weekly loop. When a course is found to end, subroutine CRSEND is called which closes out the course, allocates the AOB, and calculates attrition. When all of the 300 possible active courses have been tested, the weekly loop recycles to run another week.

Every 13 weeks, the quarter loop cycles, calling subroutine QRTEND, to allocate AOB against quarters and accumulate totals. Finally, after 4 quarters, the year loop cycles, calling subroutine YEARND, which controls printout for the yearly report.

Scheduling Algorithm. Several scheduling algorithms were studied as candidates for this model. As part of the design effort, the actual convening dates for many of the courses were plotted. These plots were then analyzed and an allowance was made for common holidays. The data were then transferred to the matrix shown in Figure VI-7. This shows the scheduled convening weeks for courses with a convening frequency of 1 to 50 per year. Many courses convened more frequently that this, and these are handled by breaking them into multiple courses, each with convening frequencies of less than 50, but whose overall convenings are the same as the original. For example, a course with 51 convenings is handled as two courses, one that convenes 25 times annually. Provision exists to break each course into 10 separate courses. The convening matrix itself is stored in core and names ISCHED. The data are loaded in hexidecimal format to accomplish this. FORTRAN allows the loading of hexidecimal data, but does not allow individual bit testing or manipulating. Therefore, a synthetic method of bit testing was developed. This technique involves extracting the proper bit pattern by division by various constants based on week and quarter, which leaves a 1 on the low order bit position if the course is to start, or 0 if it should not. This extracted bit pattern is then tested to determine whether the low order bit is a 0 or 1, which is accomplished by a division by integer 2, and then a multiplication by integer 2. If the result

is the same as the original, the low order bit was 0 and the course is not released. Two rationalizations must be made prior to lookup in this table. First, if a course convenes only once a year, the matrix will start it the first week of the model run. In reality, this course may convene on the 21st week. Offset was introduced as an input to the model to allow fine tuning of the course release date. An offset of 20, for example, will tell the model to enter the schedule matrix as week 1 when the model is in the 21st week, the second term to be rationalized. If the schedule change date has passed, the "B" convening schedule is used, otherwise the "A" convenings are used.

As this technique uses matrix lookup, the table must be protected for invalid entries while preserving flexibility. If the convening frequency is 0, the matrix will be bypassed. If the week is greater than 52, 52 is subtracted from the week until it is less than 52. If offset is positive, such as 21, no courses will be released until the 22nd week, regardless of the convening frequency. A negative offset of 31 would accomplish the same schedule shift, and courses could be released from the start of the model run.

SUBROUTINE CRSRLS. The course release subroutine contains the scheduling logic for the center. This includes keeping backlog counts, scheduling only local seats, and overbooking certain courses with high no-show rates. The logic of this subroutine schedules the BUPERS students based on the annual BUPERS plan, divided by current convening frequency. This is done even if the result exceeds the seats controlled by BUPERS. Next, local quotas are released. This is accomplished by releasing students to a backlog pool based on local demand, divided by current convenings. This pool is released to the course up to the local capacity. No-shows are then calculated using historical no-show percentages. Only local students are used for this calculation as BUPERS students generally are on TAD or PCS orders, and are not identifiable as no-shows. Finally, for courses running near capacity, up to 10 percent of the capacity is released from the backlog pool as substitute quotas. This routine then builds an entry in the course active matrix. Included is utilization, weeks till termination, and a term to allocate AOB between quarters and between classes if a student should be set back. This subroutine closely represents the scheduling methods in use at the center. If other centers, with different algorithms, should be studied, this subroutine would most likely be replaced by one representing the location under study.

SUBROUTINE PRERUN. As mentioned previously, the model can run in an initialization mode for one quarter prior to the actual model run. Subroutine PRERUN is called by CRSRLS any time the model is in the initialization mode. It is required for two functions. First, courses are tested prior to release to ascertain whether they should be running at model start or not at all. Secondly, this subroutine allocates AOB for those courses to assure that AOB applicable to the first quarter is charged to the first quarter.

SUBROUTINE CRSEND. This subroutine is called each time a course is found to end in a particular week. Several functions are performed by this subroutine. The first function consists of disposing of all references to the course in the active matrix. Secondly, failures are calculated, with fractional failures pooled, using the current failure rate. The third function is to calculate non-academic disenrollments, based on historical rates. This subroutine also

calculates setbacks. Setback logic currently allows setbacks for those courses 3 weeks or longer, and where more than 1 convening is in session at a time. This setback rate is currently based on a fixed percentage of failures per class. AOB for setback is allocated between the "entered in" and "finished in" convening. Finally, this subroutine calculates AOB based on the current calculation method in effect.

SUBROUTINE QTREND. This short subroutine is used to accurately allocate AOB between quarters. Only the active courses are tested at the end of a quarter, the accumulated AOB is charged to the current quarter, and the active course AOB is adjusted accordingly.

SUBROUTINE YEARND. The function of subroutine YEARND is to calculate annual totals for all courses, prepare the model for multiple year runs, aggregate school and center total if those reports are keyed, and call appropriate subroutines to provide output reports. This subroutine requires that courses be entered in the data matrix sorted by school code if proper school summary reports be produced.

SUBROUTINE CRTOUT. Subroutine CRTOUT produces course, school, and center reports in a 79 column format. This format is compatible with the Hazeltine 2000 terminal which only has 79 usable columns for formatted data. A sample of the reports produced by subroutine CRTOUT is shown in Figure VI-8. The format used for the control character in column 1 was described in this section under display support programs. The same format routines are used for the data portion of the school and center reports, and subroutine YEARND loads center data into the school matrix prior to calling CRTOUT to provide a center report.

COURSE SCHEDULER MATRIX

TPF MODEL COURSE RELEASE SCHEDULER

ANNUAL CONV	*** 1ST QTR *** (WKS 1-13)	*** 2ND QTR *** (WKS 1-13)	*** 3RD QTR *** (WKS 1-13)	*** 4TH QTR *** (WKS 1-13)
1	X ---- ---- ----	- ---- ---- ----	- ---- ---- ----	- ---- ---- ----
2	X ---- ---- ----	- ---- ---- ----	- X--- ---- ----	- ---- ---- ----
3	X ---- ---- ----	- ---X ---- ----	- ---- ---- -X--	- ---- ---- ----
4	X ---- ---- ----	X ---- ---- ----	- X--- ---- ----	X ---- ---- ----
5	X ---- ---- -X--	- ---- --X- ----	- ---- -X-- ----	- --X- ---- ----
6	X ---- ---- X---	- ---X ---- ----	- X--- ---- -X--	- ---- X--- ----
7	- X--- ---X ----	- -X-- ---- X---	- ---- X--- ---X	- ---- -X-- ----
8	X ---- --X- ----	X ---- -X-- ----	- X--- ---X ----	- X--- --X- ----
9	- X--- -X-- ---X	- ---X ---- -X--	- ---X ---- -X--	- -X-- ---X ----
10	X ---- X--- -X--	- -X-- --X- ----	- X--- -X-- --X-	- --X- ---X ----
11	X ---- X--- -X--	- -X-- -X-- -X--	- ---X ---- X---	- X--- X--- X---
12	- X--- X--- X---	X ---X ---X ----	- X--- -X-- -X--	- X--- X--- X---
13	- X--- X--- X--X	- --X- --X- --X-	- ---X ---X --X-	- -X-- -X-- -X--
14	X ---X ---X --X-	- -X-- X--- X---	- -X-- X--- X--X	- --X- -X-- -X--
15	X --X- --X- -X--	X ---X --X- -X--	- --X- -X-- X---	X --X- -X-- -X--
16	X --X- --X- -X--	X --X- -X-- X---	- X--- X--X --X-	- X--X --X- -X--
17	- X--X --X- -X-X	- -X-- X--X --X-	- --X- -X-- X--X	- -X-- X--X --X-
18	X --X- -X-- X--X	- -X-- X-X- -X--	- -X-- X--X --X-	- X-X- -X-- X-X-
19	- X-X- X--- X-X-	- X-X- -X-X --X-	- --X- -X-X --X-	- X-X- -X-- X--X
20	X --X- X--X -X--	X -X-- X-X- -X--	- X--X -X-- X-X-	- X-X- -X-X --X-
21	- X-X- X--X -X-X	- -X-X -X-- X-X-	- -X-- X-X- -X-X	- X--X -X-X --X-
22	- X-X- X--X -X-X	- -X-X -X-X -X--	- -X-X -X-X -X-X	- -X-X --X- X-X-
23	- X-X- X-X- X-X-	- X-X- X-X- X-X-	- ---X X-X- X-X-	- X-X- X-X- X-X-
24	- X-X- X-X- X-X-	X -X-X -X-X -X--	- -X-X -X-X -X-X	- X-X- X-X- X-X-
25	X -X-X -X-X -X-X	- X-X- X-X- X-X-	- -X-X -X-X -X-X	- X-X- X-X- X-X-
26	X X-X- X-X- X-X-	X -X-X -X-X -X--	- X-X- X-X- X-X-	X -X-X -X-X -X-X
27	X X-X- X-X- X-X-	X -X-X -XX- X-X-	- X-X- X-XX -X-X	- X-X- X-X- X-X-
28	X -X-X -XX- X-X-	X X-X- X-X- X-X-	- X-X- X-X- X-X-	X X-X- XX-X -X-X
29	X -X-X -XX- X-X-	X X-X- X-XX -X--	- X-XX -X-X X-X-	X -XX- X-X- XX-X
30	- XX-X -XX- X-XX	- X-XX -X-X X-X-	- XX-- X-XX -X-X	X -X-X X-X- XX-X
31	X -X-X X-XX -X-X	X -X-X X-X- XX--	- XX-X X-X- XX-X	X -X-X X-XX -XX-
32	- XX-X X-XX -XX-	X X-XX -X-X X-X-	- X-XX -XX- XX-X	X -X-X X-XX -XX-
33	X -XX- XX-X X-X-	X X-XX -XX- XX--	- XX-X X-XX -XX-	X X-XX -XX- XX-X
34	- XX-X XX-X X-XX	- XX-X X-XX -XX-	- -XXX -XX- XX-X	X -XX- XX-X X-XX
35	- XX-X XX-X X-XX	- XXX- XX-X X-X-	- XX-X X-XX -XXX	- XX-X X-XX X-XX
36	- XXX- XXX- XX-X	X X-XX -XXX -XX-	- X-XX -XXX -XX-	X XX-X X-XX X-XX
37	X -XXX -XXX -XX-	X XX-X XX-X XX--	- XXX- XXX- XX-X	X X-XX X-XX X-XX
38	X -XXX -XXX -XXX	- XXX- XXX- XXX-	- -XXX X-XX X-XX	X -XXX -XXX -XXX
39	- XXXX -XXX X-XX	X X-XX X-XX X-X-	- XXX- XXXX -XXX	X -XXX -XXX -XXX
40	- XXXX -XXX X-XX	X X-XX XX-X XXX-	- -XXX X-XX XX-X	X XX-X XXX- XXXX
41	X -XXX X-XX XXX-	X XXX- XXXX X-X-	- XXX- XXXX X-XX	X X-XX XXX- XXXX
42	- XXXX XX-X XXXX	- XXXX X-XX XXX-	- -XXX XXX- XXXX	X -XXX XX-X XXXX
43	X -XXX XXX- XXXX	X X-XX XXXX -XX-	- XXXX -XXX XXX-	X XXXX X-XX XXXX
44	- XXXX XXXX -XXX	X XXX- XXXX XXX-	- XXXX -XXX X-XX	X XXXX -XXX XXXX
45	- XXXX XXXX X-XX	X XXXX XX-X XXX-	- XXXX X-XX XXXX	X XX-X XXXX XXXX
46	- XXXX XXXX XXXX	- XXXX XXXX XXX-	- -XXX XXXX XXXX	- XXXX XXXX XXXX
47	- XXXX XXXX XXXX	X XXX- XXXX XXX-	- XXXX XXXX X-XX	X XXXX XXXX XXXX
48	- XXXX XXXX XXXX	X XXXX XXXX XXX-	- -XXX XXXX XXXX	X XXXX XXXX XXXX
49	- XXXX XXXX XXXX	X XXXX XXXX XXX-	- XXXX XXXX XXXX	X XXXX XXXX XXXX
50	X XXXX XXXX XXXX	X XXXX XXXX XXX-	- XXXX XXXX XXXX	X XXXX XXXX XXXX

FOR EACH WEEK IN THIS SCHEDULE: AN 'X' INDICATES THE COURSE WILL BE RELEASED
A '-' INDICATES THE COURSE WILL NOT BE RELEASED

NOTE: IF ANNUAL CONVENINGS FOR A COURSE IS GREATER THAN 50, THE COURSE WILL BE TREATED AS MULTIPLE COURSES, EACH HAVING 50 OR LESS CONVENINGS PER YEAR.

FIGURE VI-7

0284	FINANCIAL MGMT			--SCHEDULE DATA--	FY	FY+1	FY+2	CRS BACKLOG			
				ANNUAL CONVENINGS	13	0	0	0 STU	32		
SCHL DATE CATALOG NO			NLC	COURSE LENGTH DYS	5.0	0.0	0.0	0 WKS	5		
S/DA 0812 A-551-0056				COURSE CAPACITY	25	0	0				
				BUPERS CAPACITY	0	0	0	FY+1 WK	0		
P AOB	PFAIL	TYPE	PRTY	ANNUAL DEMAND	150	180	180	FY+2 WK	0		
2.0	7.0%	F1	31	BUPERS DEMAND	0	0	0	OFFSET	0		

QUARTERLY PROJECTIONS

	1 QR FY 75		2 QR FY 75		3 QR FY 75		4 QR FY 75		ANNUAL TOTAL	
0 CONVENINGS	4		3		3		3		13	
CRS CAPACITY	100		75		75		75		325	
UTILIZATION	67	67.0%	30	40.0%	29	38.7%	30	40.0%	156	48.0%
NO SHOWS	11	14.1%	5	14.3%	5	14.7%	5	14.3%	26	14.3%
BACKLOG 0/WKS	0	0 WKS	0	0 WKS	0	0 WKS	0	0 WKS	0	0 WKS
STU DAYS/AOB	335	3.7	150	1.6	145	1.6	150	1.6	780	2.1
TOTAL PASS	62	92.5%	27	90.0%	25	86.2%	26	86.7%	140	89.7%
TOTAL FAIL	5	7.5%	3	10.0%	3	10.3%	3	10.0%	14	9.0%
SETBACK ACAD	0	0.0%	0	0.0%	0	0.0%	0	0.0%	0	0.0%
DISENROLL NA	0	0.0%	0	0.0%	1	3.4%	1	3.3%	2	1.3%

TPF MODEL -- FISCAL YEAR 1976 SUMMARY
SCHOOL CODE COMM

COMM SCHOOL SUMMARY (RUN 081275)

	1 QR FY 76		2 QR FY 76		3 QR FY 76		4 QR FY 76		ANNUAL TOTAL	
0 CONVENINGS	47		41		44		41		173	
CRS CAPACITY	683		589		645		589		2506	
UTILIZATION	602	88.1%	447	75.9%	441	68.4%	383	65.0%	1873	74.7%
NO SHOWS	77	11.3%	68	13.2%	64	12.7%	58	13.2%	267	12.5%
BACKLOG 0/WKS	181	7 WKS	90	6 WKS	46	5 WKS	27	3 WKS	27	3 WKS
STU DAYS/AOB	2810	30.8	2052	22.5	2051	22.5	1661	18.2	8574	23.5
TOTAL PASS	588	97.7%	434	97.1%	430	97.5%	372	97.1%	1824	97.4%
TOTAL FAIL	14	2.3%	13	2.9%	11	2.5%	11	2.9%	49	2.6%
SETBACK ACAD	0	0.0%	0	0.0%	0	0.0%	0	0.0%	0	0.0%
DISENROLL NA	0	0.0%	0	0.0%	0	0.0%	0	0.0%	0	0.0%

TPF MODEL -- FISCAL YEAR 1976 SUMMARY
** TOTALS FOR TRAPAC TRAINING CENTER **

CENTER CODE N SUMMARY (RUN 081275)

	1 QR FY 76		2 QR FY 76		3 QR FY 76		4 QR FY 76		ANNUAL TOTAL	
0 CONVENINGS	759		695		678		720		2852	
CRS CAPACITY	23900		22514		21648		24265		92335	
UTILIZATION	16209	67.8%	12559	55.8%	11825	54.6%	12918	53.2%	53511	58.0%
NO SHOWS	2423	13.0%	1974	13.6%	1861	13.6%	2082	13.9%	8340	13.5%
BACKLOG 0/WKS	3432	9 WKS	2676	11 WKS	2325	13 WKS	2172	15 WKS	2172	15 WKS
STU DAYS/AOB	92066	1000.9	73855	809.4	71520	783.8	73931	810.2	311372	853.1
TOTAL PASS	15686	96.8%	12869	96.1%	11348	96.0%	12393	95.9%	51496	96.2%
TOTAL FAIL	417	2.6%	375	3.0%	365	3.1%	400	3.1%	1557	2.9%
SETBACK ACAD	36	0.2%	27	0.2%	27	0.2%	38	0.2%	128	0.2%
DISENROLL NA	106	0.7%	115	0.9%	112	0.9%	125	1.0%	458	0.9%

SAMPLE TPF OUTPUTS

FIGURE VI-8

KEND FORTRAN

```
C ************** TPF MODEL - AS OF TUESDAY FEB 24, 1976 ***************
      IMPLICIT INTEGER*2 (A-Z,$)
      INTEGER*2 DOLLAR /'$'/
      REAL*4 SELCRS, TCDP
C COMMON DATA FOR CLASS DATA MATRIX
C
      REAL*4 LEN, NOSHOW, NOSHPL, BKLGPL, BPOOL, NADIS, FAIL, CDP
      REAL*4 TDMAND, SETBAK, DATE
C
      COMMON /DATA/ CDP (328), CRSRUN (328), CONV (3,328), LEN (3,328)
      COMMON /DATA/ CAP (3,328), BCAP (3,328), CHGWK (2,328)
      COMMON /DATA/ OFFSET (328), DEMAND (3,328), BDMAND (3,328)
      COMMON /DATA/ BPOOL (328), FAIL (328), NADIS (328), NOSHPL (328)
      COMMON /DATA/ BAKLOG (328), NOSHOW (328), BKLGPL (328)
      COMMON /DATA/ BKLG (328), TDMAND, SETBAK, DATE
C
      INTEGER*4 PNEC
      REAL*4 PAOB, PFAIL
      REAL*8 NAME1, NAME2, CATNO1
C
      COMMON /TEXT/ PFAIL(328), NAME1 (328), NAME2 (328), CATNO1 (328)
      COMMON /TEXT/  CATNO2 (328), PNEC (328), TYPE (328), PAOB (328)
      COMMON /TEXT/ PRTY(328),SCH1(328),SCH2(328),SCH3(328),SCH4(328)
C
C QUARTERLY AND ANNUAL COURSE TOTALS ACCUMULATION
C
      REAL*4 TFAIL, TNADIS
      INTEGER*4 TSDAYS, TCAP, TUTIL
C
      COMMON /TOTALS/ TCAP (5, 328),   TUTIL (5, 328),  TSETBK (5, 328)
      COMMON /TOTALS/ TSDAYS (5, 328), TFAIL (5, 328),  TNOSHO (5, 328)
      COMMON /TOTALS/ TNADIS (5, 328), TBKLOG (5, 328), TCONV (5, 328)
C
C COMMON FOR COURSE ACTIVE MATRIX ELEMENTS
C
      REAL*4 ACTCTR
      COMMON /RUN/ ACTIVE (250), ACTCTR (250), ACTUTL (250), ACTNEG(250)
C
C COMMON FOR COMMUNICATION BETWEEN SUBROUTINES
      INTEGER*4 ITEM, ITEM1, ITEM2, ITEM3, ITEM4, ITEM5, WHERE
      REAL*4 TEM, TEM1, TEM2, TEM3, TEM4, TEM5, $LEN
C
      COMMON DUMMY, $CONV, $LEN, $CAP, $BCAP, LAST, COURSE
      COMMON $R1,$R2,$R3,$R4, TEM, TEM1, TEM2, TEM3, TEM4, TEM5
      COMMON ITEM, ITEM1, ITEM2, ITEM3, ITEM4, ITEM5
      COMMON YEAR, CRSES, G, W, WKOFF, INIT, WHERE, NEGFLG, AOBCON
      COMMON TRACEF, RLSFLG, ENDFLG, PREFLG, YRFLG,YEARUN,PRINTF,NGR
C
C DIMENSION AND DATA FOR SPECIAL SCHEDULER ROUTINE
C
      DIMENSION C (10), ISCHD (4, 50), DIV (13)
C
      DATA ISCHD/Z800,3*Z0,Z0,Z800,Z2,Z0,Z0,Z1000,Z100,Z4,2*Z1000,
     *Z800,Z1000,Z8,Z40,Z100,Z801,Z100,Z1008,Z101,Z10,Z810,Z408,Z81,
     *Z40,Z1020,Z1040,Z810,Z820,Z841,2*Z104,Z410,Z1084,Z420,Z842,Z210,
```

KEND FORTRAN

```
     *Z1084,Z444,Z106,2*Z888,Z1110,Z844,Z888,Z889,Z222,Z112,Z444,Z1112,
     *Z488,Z489,Z244,Z1224,Z1124,Z248,Z1244,Z1224,Z1248,Z892,Z924,Z925,
     *Z492,Z249,Z492,Z1249,Z4A4,Z492,ZA4A,ZA8A,ZA52,Z252,ZA49,Z1294,
     *Z14A4,Z94A,ZA52,Z495,Z54A,Z4A5,Z952,ZA95,Z554,Z555,Z52A,2*ZAAA,
     *Z1AA,2*ZAAA,Z1554,Z555,ZAAA,Z1555,ZAAA,Z555,ZAAA,Z1AAA,Z1554,ZAAA,
     *Z1555,Z1AAA,Z156A,ZAB5,ZAAA,Z156A,Z1AAA,ZAAA,Z1AD5,Z156A,Z1AB4,
     *ZB5A,Z16AD,ZD6B,ZB5A,Z6B5,Z15AD,Z15B5,Z15AC,ZDAD,Z15B6,ZDB6,Z1B5A,
     *ZB6D,Z15B6,Z16DA,Z1B6C,ZDB6,Z1B6D,ZDDB,ZDB6,Z76D,Z16DB,ZDDB,ZEDA,
     *ZDB7,ZDBB,ZEED,Z1B76,ZB76,Z1DBB,Z1776,Z1DDC,ZEED,Z1BBB,Z1777,ZEEE,
     *Z7B5,Z1777,ZF7B,Z1BBA,ZEF7,Z1777,ZF7B,Z1BDE,Z7BD,Z1DEF,Z17BE,
     *Z1EFA,ZEFB,Z1BEF,ZFDF,ZFBE,Z7EF,Z17DF,Z17EF,Z1BF6,ZF7E,Z1FBF,ZFF7,
     *Z1EFE,ZF7B,Z1F7F,ZFFB,Z1FDE,ZFBF,Z1DFF,ZFFF,ZFFE,Z7FF,2*ZFFF,
     *Z1EFE,ZFFB,Z1FFF,ZFFF,Z1FFE,Z7FF,Z1FFF,ZFFF,Z1FFE,ZFFF,2*Z1FFF,
     *Z1FFE,ZFFF,Z1FFF/
C
      DATA DIV/4096,2048,1024,512,256,128,64,32,16,8,4,2,1/
      DEFINE FILE 10 (800,80,E,J3)
      EXTERNAL RANGE
      CALL ERRSET (232,100,-1,1,RANGE)
C
C READ RUN PARAMETERS:  'WHERE' IS INPUT DEVICE; 'FRSTYR & LASTYR' ARE
C FIRST AND LAST FISCAL YEARS OF RUN.
C OVERRIDES - ITEM(FAIL), ITEM1(NOSHOW), ITEM2(DEMAND), ITEM3(CONV),
C ITEM4(LEN), ITEM5(CAP), IDMAND IS DEMAND TREND OVERRIDE - USED YREND
C Q FLG FOR COURSE SEL, W FLG FOR SCHOOL SEL, I2 FLG FOR LENGTH IN WEEKS
C
      READ (5, 1000, END=1998) WHERE, FRSTYR, LASTYR, AOBCON,
     1 ITEM, TEM, ITEM1, TEM1, ITEM2, TEM2, I, IDMAND, ITEM3, TEM3,
     2 ITEM4, TEM4, ITEM5, TEM5,DATE,I1,SETBAK, Q, SELCRS, W, SLSCH1,
     3 SLSCH2,SLSCH3,SLSCH4,YRFLG,WKFLG,TRACEF,PREFLG,RLSFLG,ENDFLG,
     4 STAT, $R1,$R2,$R3
 1000 FORMAT(I2, 3I4, 1X, / (I1, F4.3),1X,A4,I1,F4.3,I1,A4,I1,4A1,10I1)
C
C CLAMP SETBACK RATE (AS PCT OF FAILURES), CLAMP AOB CONST IF MISSING
      IF (I1 .LE. 0) SETBAK = 0.5
      IF (AOBCON .LE. 0) AOBCON = 365
      IF (I .LE. 0) IDMAND = 0
C
C SET INIT FLAG - USED TO KEY MODEL PRERUN
C CLAMP INIT AS USED TO SET NEGRUN FLAG WHOSE ACTUAL VALUE IS SIGNIFCANT
C
      INIT = 1 ; PRINTF = 0 ; NEGFLG = INIT
C
C SET TRACE FLAG - USED FOR DETAILED COURSE RUN PRINTOUT
      TRACEF = TRACEF + PREFLG + RLSFLG + ENDFLG
C
C CLAMP DEVICE TO READER OR DISK FOR DATA INPUT
C
      IF (WHERE .NE. 10) WHERE = 5
      IF ($R2 .GT. 1) $R2 = 1
      NGR = 0
C
C
C FIND NUMBER OF YEARS FOR MODEL RUN - MAX 3 YEARS, PROTECT FOR NEGATIVE
C FRSTYR IS FIRST FY OF RUN, LASTYR IS LAST YR, YEAR IS CURRENT YEAR OF
```

KEND FORTRAN

```
C RUN. ASSUME 1976 IF MISSING. INVERT IF YRS OUT OF ORDER
C
C TEST FOR MISSING DATA
      YEARUN = 1
      IF (FRSTYR) 26, 26, 27
C
C DATA MISSING-ASSUME 1976 & RUN 1 YEAR
   26 YEAR = 1976 ; GO TO 32
C
C HAVE FRST YEAR, TEST FOR LAST. RUN 1 YEAR IF NONE
   27 IF (LASTYR) 31, 31, 28
C
C HERE IF HAVE 1ST & LAST YR. FIND DELTA FOR YEARS RUN
   28 YEARUN = LASTYR - FRSTYR
      IF (YEARUN) 29, 30, 30
C
C NEG DELTA. YEARS OUT OF ORDER - STRAIGHTEN OUT
   29 YEARUN = -YEARUN ; FRSTYR = LASTYR
   30 YEARUN = YEARUN + 1
   31 YEAR = FRSTYR
      IF (YEARUN .GT. 3) YEARUN = 3
   32 CONTINUE
      SELYR = 1
      IF (FRSTYR .EQ. 1977) SELYR = 2
      IF (FRSTYR .EQ. 1978) SELYR = 3
C
C PRINT CHARACTERISTICS UNIQUE TO THIS MODEL RUN
C
      WRITE (6, 1001)
 1001 FORMAT ('1BEGIN TRAINING PROCESS FLOW MODEL EXECUTION - CHARACTERI
     1STICS FOR THIS RUN ARE:')
C
      WRITE (6, 1002) YEAR, YEARUN
 1002 FORMAT (/ ' FIRST FISCAL YEAR OF MODEL RUN', I5, 4X, 'YEARS PROJEC
     *TED', I3)
C
      IF (AOBCON .GT. 300) WRITE (6, 1006) AOBCON
 1006 FORMAT (/' AOB CONSTANT IS', I4, '  WEEKEND DAYS USED IN STUDENT D
     *AYS AND AOB CALC')
C
      IF (AOBCON .LE. 300) WRITE (6, 1007) AOBCON
 1007 FORMAT (/' AOB CONSTANT IS', I4, ' ONLY WORKING DAYS USED IN STUD
     *ENT DAYS AND AOB CALC')
C
      IF (WHERE .EQ. 5) WRITE (6, 1008)
 1008 FORMAT (/ ' COURSE DATA INPUT DEVICE - READER:    STATISTICS INPUT
     * DEVICE - READER')
C
      IF (WHERE .GE. 10) WRITE (6, 1009)
 1009 FORMAT(/' ALL INPUTS THIS RUN FROM UNLOADED DATA BASE')
C
      WRITE (6, 1010) PREFLG, RLSFLG, ENDFLG, YRFLG
 1010 FORMAT(/' TRACE FLAGS THIS RUN:  PRERUN', I2, '; CRS REL', I2, '; 
     *CRS END', I2, '; YEAR FORMAT', I2 /)
C
```

TABLE VI-17 (Continued)

KEND FORTRAN

```
      IF (INIT .GT. 0) WRITE (6, 1012)
 1012 FORMAT ( ' INIT FLAG IS SET - MODEL WILL INIT COURSES IN SESSION A
     *T 1ST QUARTER START' /)
C
      IF (STAT .GT. 0) WRITE (6, 11013)
11013 FORMAT ( ' STAT FLAG IS SET-STATISTICAL FAIL DATA WILL OVERRIDE HI
     *STORICAL RATES' /)
C
C END OF MODEL RUN DATA MESSAGES - OUTLINE ERROR MESSAGES
C
      WRITE (6, 1011)
 1011 FORMAT ( ' ------------------------------------------------------
     *-------------------------')
C CDP OR SCHOOL SELECT & READ ROUTINE
      DONEFL = 0 ;/* INIT DONE FLAG FOR SELECT OPT */
      IF (Q .LE. 0) GO TO 2120            ;/* SELECT CDP OPTION */
      WRITE (6,12014 )SELCRS
12014 FORMAT (' CDP SELECT= ', A4)
C
      DO 2117 I1 = 1, 656, 2
      READ (10'I1,2111,ERR=2118) TCDP
 2111 FORMAT (4X,A4,72X)
      IF (NGR .GT. 0) GO TO 2118
      IF (TCDP .EQ. SELCRS) GO TO 2119
 2117 CONTINUE
 2118 GO TO 1998  ;/*EXIT- NO CRSES */
 2119 DONEFL = 1 ; GO TO 2131
 2120 IF (W .LE. 0) GO TO 2130  ;/* SELECT SCHOOL OPTION */
      I1 = 1 ; IF (SLSCH1 .EQ. DOLLAR) GO TO 2129
      IF ($R2 .GT. 0) $R2 = 2 ;/* KEY SEL SCH- 2 IFSELECT ON */
      DO 2127 I1 = 1,656,2
      READ (10'I1,2121,ERR=2128)TSTSCH
 2121 FORMAT (A1, 79X)
      IF (NGR .GT. 0) GO TO 2128
      IF (TSTSCH .EQ. SLSCH1) GO TO 2129
 2127 CONTINUE
 2128 GO TO 1998  ;/* EXIT- NO CRSES */
 2129 DONEFL = -1 ; GO TO 2131  ;/* SET FLAG- GO READ */
C READ IN DATA- DONEFL = -1 IF SCHL SEL, +1 IF CDPSEL, 0 IF NONE
 2130 I1 = 1
 2131 CRSES = 0
 2132 I = CRSES + 1
      READ(10'I1,2133,ERR=2139)SCH1(I),SCH2(I),SCH3(I),SCH4(I),CDP(I),
     1 CATNO1(I), CATNO2(I), NAME1(I), NAME2(I), CONV(1,I), LEN(1,I),
     2 CAP (1,I), BCAP (1,I), DEMAND(1,I), BDMAND(1,I), OFFSET(I),
     3 FAIL(I), NADIS (I), NOSHOW (I), BAKLOG (I),
     4 CONV (2,I), LEN (2,I), CAP (2,I), BCAP (2,I), DEMAND (2,I),
     5 BDMAND (2,I), CHGWK(1,I), CONV (3,I), LEN (3,I), CAP (3,I),
     6 BCAP (3,I), DEMAND (3,I), BDMAND (3,I), CHGWK (2,I), PAOB (I),
     7 PFAIL (I), PRTY (I), TYPE (I), PNEC (I)
C
 2133 FORMAT (4A1,A4,1X,A8,A2,2A8,I3,F4.0,2I3,2I5,I2,3F5.3,3X, /
     * I4,4X, 2(I3,F4.0,2I3,2I5,I3), F4.1,F5.3,2A2,A4,3X)
C
      IF (NGR .GT. 0) GO TO 2139
```

TABLE VI-17 (Continued)

KEND FORTRAN

```
      I1 = I1 + 2
      IF (DONEFL) 2135, 2134, 12139 ; /* -1 SEL,0 NORM,1 CDP*/
 2134 CRSES = CRSES + 1 ; IF (I1 .GT. 656) GO TO 2139 ;/*EXIT- FULL */
      GO TO 2132  ;/*READ ANOTHER COURSE */
C SCHOOL SELECT - DOLLAR IN FIELD BLANKS SEARCH
 2135 IF (SLSCH1 .EQ. DOLLAR) GO TO 2136
      IF (SLSCH1 .EQ. SCH1(I)) GO TO 2136
      GO TO 2139
 2136 IF (SLSCH2 .EQ. DOLLAR) GO TO 2137
      IF (SLSCH2 .EQ. SCH2(I)) GO TO 2137
      GO TO 2132
 2137 IF (SLSCH3 .EQ. DOLLAR) GO TO 2138
      IF (SLSCH3 .EQ. SCH3(I)) GO TO 2138
      GO TO 2132
 2138 IF (SLSCH4 .EQ. DOLLAR) GO TO 2134
      IF (SLSCH4 .EQ. SCH4(I)) GO TO 2134
      GO TO 2132
12139 CRSES = CRSES + 1
 2139 IF (CRSES .LE. 0) GO TO 1998 ;/* EXIT- NO CRSES */
C
      WRITE (6, 12002) CRSES
12002 FORMAT (' COURSES ACTIVE THIS RUN', I5)
C
CC INITIALIZE BACKLOG POOL - BAKLOG IS COURSE BACKLOG IN WEEKS, CONVERT
C TO STUDENT DEMAND & INIT TO BKLGPL. ZERO BACKLOG IF NEGATIVE INPUT ERR
C INIT BACKLOG POOL TO B CONVENINGS IF NO 'A' CONVENINGS.
C
      DO 720 I = 1, CRSES
      IF (BAKLOG (I))711, 720, 712
  711 BAKLOG (I) = 0
      WRITE (6, 11011) CDP (I)
11011 FORMAT (' NEG BACKLOG COURSE ', A4, '  ZEROED THIS RUN')
      GO TO 720
C
  712 I1 = 1
      IF (CONV (I1, I)) 713, 713, 714
C NO 'A' CONVENINGS, TEST B CONVENINGS
  713 I1 = 2
      IF (CONV (I1, I)) 720, 720, 714
C HAVE CONVENINGS, I1 POINTS AT WHICH ONE
  714 I2 = CAP (I1, I) - HCAP (I1, I)
      IF (I2) 720, 720, 715
  715 SELCRS = 50. / CONV (I1, I)
      BKLGPL (I) = (BAKLOG (I) / SELCRS) * I2
      BKLG (I) = BKLGPL (I)
  720 CONTINUE
C
C OVERRIDE ROUTINES - FROM PARAMETER CARD
C
C FAIL OVERRIDE - ITEM IS FLAG, TEM IS MULTIPLIER
C
      IF (ITEM) 91007, 91007, 91001
91001 IF (TEM) 91002, 91004, 91004
91002 WRITE (6, 91003) TEM
91003 FORMAT (' NEGATIVE OVERRIDE - REQUEST IGNORED, VALUE =', F6.3)
```

TABLE VI-17 (Continued)

KEND FORTRAN

```
      GO TO 91007
91004 WRITE (6, 91005) TEM
91005 FORMAT (' FAIL OVERRIDE- RATES MULT BY', F6.3, ' THIS RUN ONLY')
      DO 91006 I = 1, CRSES
91006 FAIL (I) = FAIL (I) * TEM
C
C NO SHOW OVERRIDE - FLAG IS ITEM1, MULT IS TEM1
C
91007 IF (ITEM1) 91017, 91017, 91011
91011 IF (TEM1) 91012, 91014, 91014
91012 WRITE (6, 91003) TEM1
      GO TO 91017
91014 WRITE (6, 91015) TEM1
91015 FORMAT (' NO SHOW OVERRIDE - RATES MULT BY',F6.3,' THIS RUN ONLY')
      DO 91016 I = 1, CRSES
91016 NOSHOW (I) = NOSHOW (I) * TEM1
C
C DEMAND OVERRIDE - FLAG IS ITEM2, MULT IS TEM2
C
91017 IF (ITEM2) 91027, 91027, 91021
91021 IF (TEM2) 91022, 91024, 91024
91022 WRITE (6, 91003) TEM2
      GO TO 91027
91024 WRITE (6, 91025) TEM2
91025 FORMAT (' DEMAND OVERRIDE-DEMANDS MULT BY',F6.3,' THIS RUN ONLY')
      DO 91026 I = 1, CRSES
      DO 91026 I1 = 1, 3
      DEMAND (I1, I) = DEMAND (I1, I) * TEM2 + .5
91026 BDMAND (I1, I) = BDMAND (I1, I) * TEM2 + .5
C
C OVERRIDE ANNUAL CONVENINGS - FLAG IS ITEM3, MULT IS TEM3
C
91027 IF (ITEM3) 91037, 91037, 91031
91031 IF (TEM3) 91032, 91032, 91034
91032 WRITE (6, 91003) TEM3
      GO TO 91037
91034 WRITE (6, 91035) TEM3
91035 FORMAT (' CONVENINGS OVERRIDE-FREQ MULT BY',F6.3,' THIS RUN ONLY')
      DO 91036 I = 1, CRSES
      DO 91036 I1 = 1, 3
91036 CONV (I1, I) = CONV (I1, I) * TEM3 + 0.5
C
C CAPACITY OVERRIDE - FLAG IS ITEM5, MULT IS TEM5
C
91037 IF (ITEM5) 91057, 91057, 91051
91051 IF (TEM5) 91052, 91052, 91054
91052 WRITE (6, 91003) TEM5
      GO TO 91057
91054 WRITE (6, 91055) TEM5
91055 FORMAT (' CAPACITY OVERRIDE-CAP MULT BY',F6.3,' THIS RUN ONLY')
      DO 91056 I = 1, CRSES
      DO 91056 I1 = 1, 3
      CAP (I1, I) = CAP (I1, I) * TEM5 + .5
91056 BCAP (I1, I) = BCAP (I1, I) * TEM5 + .5
C
```

TABLE VI-17 (Continued)

KEND FORTRAN

```
91057 IF (TDMAND .GT. 0.) WRITE (6,9105H) TDMAND
91058 FORMAT (' TREND OPTION- DEMANDS MULT BY', F6.3, ' EACH YEAR')
C
C TEST IF LENGTH INPUT IN DAYS, CONVERT TO WEEKS IF IN DAYS FOR EXECUTE
C
      IF (WKFLG) 91061, 91061, 91069
91061 DO 91068 J = 1, 3
      DO 91068 I = 1, CRSES
      IF (LEN (J,I)) 91068, 91068, 91062
91062 ITEM = LEN (J,I) / 7
      TEM = LEN (J,I) - ITEM * 7
      IF (TEM .GT. 5) TEM = 5
      LEN (J,I) = ITEM + TEM / 5.
91068 CONTINUE
C
C LENGTH OVERRIDE - MIN CHANGE - HALF DAYS ONLY
C SET TEM TO 1 IF LENGTH INPUT IN DAYS, 10 IF IN WEEKS
C
91069 IF (ITEM4) 91047, 91047, 91041
91041 IF (TEM4) 91042, 91042, 91044
91042 WRITE (6, 91003) TEM4
      GO TO 91047
91044 WRITE (6, 91045) TEM4
91045 FORMAT (' LENGTH OVERRIDE-LEN MULT BY',F6.3,' THIS RUN ONLY')
91049 DO 91046 I = 1, CRSES
      DO 91046 I1 = 1, 3
      IF (LEN (I1,I) .LE. 0) GO TO 91046
      ITEM = LEN (I1,I) * TEM4 * 10. + 1
      LEN (I1,I) = ITEM / 10.
91046 CONTINUE
C
C TEST COURSE DATA MATRIX FOR INVALID OR NEGATIVE DATA
C
91047 DO 25 I = 1, CRSES
C
      PFAIL (I) = FAIL (I) * 100.
C
C CLAMP OFFSET AT 1 YEAR
C
      ITEM = OFFSET (I)
      IF (IABS (ITEM) .LE. 51) GO TO 12
      OFFSET (I) = 0
      WRITE (6, 10017) COP (I)
10017 FORMAT (' OFFSET GT 1 YEAR FOR COURSE ', A4, '  ZEROED THIS RUN')
   12 CONTINUE
C
C TEST MULTI DIMENSIONED DATA, TEST ALL FOR I1 = 1, REST ONLY IF CHG WK
C
      DO 25 I1 = 1, 3
      I2 = I1 - 1
      IF (I2) 15, 15, 13
C
C TEST CHANGE WEEK FOR NEG, SKIP FURTHER TESTS IF NO CHANGE WEEK
   13 IF (CHGWK (I2, I)) 14, 25, 15
   14 CHGWK (I2, I) = 0
```

TABLE VI-17 (Continued)

KEND FORTRAN

```
      WRITE (6, 11035) CDP (I)
11035 FORMAT (' NEGATIVE CHANGE WEEK COURSE ',A4,' ZEROED THIS RUN')
      GO TO 25
C
C TEST CONVENINGS - BYPASS FURTHER TESTS IF 0 CONVENINGS
   15 IF (CONV (I1, I)) 16, 25, 17
   16 CONV (I1, I) = 0
      WRITE (6, 10012) CDP
10012 FORMAT (' NEGATIVE CONVENINGS FOR COURSE ', A4,' ZEROED THIS RUN')
      GO TO 25
C
   17 IF (LEN (I1, I)) 18, 19, 19
   18 LEN (I1, I) = 0.
      WRITE (6, 33) CDP
   33 FORMAT (' NEG LENGTH COURSE ', A4, '  ZEROED THIS RUN')
C
   19 IF (CAP (I1, I)) 20, 21, 21
   20 CAP (I1, I) = 0
      WRITE (6, 34) CDP (I)
   34 FORMAT (' NEGATIVE CAPACITY COURSE ', A4, '  ZEROED THIS RUN')
C
   21 IF (BCAP (I1, I)) 22, 23, 23
   22 BCAP (I1, I) = 0
      WRITE (6, 34) CDP (I)
C
   23 IF (DEMAND (I1, I)) 24, 701, 701
   24 DEMAND (I1, I) = 0
      WRITE (6, 11018) CDP (I)
11018 FORMAT (' NEGATIVE DEMAND FOR COURSE ', A4, '  ZEROED THIS RUN')
C
  701 IF (BDMAND (I1, I)) 702, 25, 703
  702 BDMAND (I1, I) = 0
      WRITE (6, 11018) CDP (I)
      GO TO 25
C
  703 IF (BCAP (I1, I))11019, 11019, 25
11019 WRITE (6, 11020) CDP (I)
11020 FORMAT (' WARNING-BUPERS DEMAND WITH NO BUPERS SEATS COURSE ', A4)
C
   25 CONTINUE
C
C
C OUTLINE ERROR MESSAGES
      WRITE (6, 1011)
C
      IF (NEGFLG .GT. 0) WRITE (6, 1013)
 1013 FORMAT(/ ' PRERUN FLAG SET- 4TH QTR WILL BE RUN TO INIT CRSES IN S
     *ESSION AT MODEL START',//)
C
C-----------------------------------------------------------------------
C ****** END MODEL INITIALIZATION - BEGIN ACTIVE MODEL LOOP ***********
C YEAR LOOP - ALLOWS MODEL TO RUN MULTIPLE YEARS, YEAR INPUT ON PARM CRD
C YEARUN HANDED OFF TO OTHER SUBROUTINES TO TEST CHANGE DATE
C-----------------------------------------------------------------------
C
```

TABLE VI-17 (Continued)

KEND FORTRAN

```
      Y = YEARUN
      DO 999 YEARUN = SELYR, Y ; /* SELYR IS 1,2, OR 3 FOR FY OF RUN */
C
C QUARTER LOOP - MODEL RUNS ON 4 QUARTERS OF 13 WEEKS
C
      DO 900 TEMQ = 1, 4
C
10800 CONTINUE ; /* ENTRY FOR PRERUN END */
      Q = TEMQ
C
C TEST FOR NEGFLAG, OVERRIDE Q IF PRERUN
      IF (NEGFLG .GT. 0) Q = 4
C
C ****** BEGIN LOWEST ORDER (WEEKLY) CYCLING FOR TPF MODEL ******
C LOOP THROUGH ENTIRE COURSE MATRIX ON A WEEKLY BASIS
C
      DO 800 W = 1, 13
C
C INITIALIZE 'LAST' TO 1, USED TO FIND LAST USED POSITION IN CRS ACTIVE
C MATRIX BY CRSRLS SUBROUTINE
C
      LAST = 1
      NOW = ((Q - 1) * 13 + W) + (52 * (YEARUN - 1))
C
C LOOP THROUGH COURSE MATRIX
C
      DO 500 I = 1, CRSES
C
C SET UP PROPER COURSE CONVENING, LENGTH, & CAPACITY DATA, DEPENDING ON
C WHETHER OR NOT A COURSE CHANGE DATE HAS PASSED
C USE A CONVENINGS IF NEGFLG (PRERUN) FLAG IS ON
C
      ITEM = 1
      IF (NEGFLG) 11034, 11034, 37
11034 IF (CHGWK (1, I)) 137, 137, 35
C
C HAVE A CHANGE WEEK - CHECK IF EITHER HAS PASSED (1 DOMINANT)
   35 IF (NOW .LT. CHGWK (1, I)) GO TO 37
      ITEM = 2
      IF (CHGWK (2,I)) 36, 36, 39
   39 IF (NOW .LT. CHGWK (2, I)) GO TO 36
C
C CHANGE WEEK 2 HAS PASSED - USE '3' DATA
      ITEM = 3
C IF CONV FREQ LT 8 SET OFFSET TO CHANGE WEEK (TO ASSURE CRS START)
   36 WKOFF = 0
      IF (CONV (ITEM, I) .LT. 8) WKOFF = CHGWK (ITEM-1, I) - 1
      GO TO 38
C
  137 ITEM = YEARUN ; /* FORCE DATA FOR YEAR OF RUN*/
   37 WKOFF = OFFSET (I)
   38 $CONV= CONV (ITEM, I)
      $LEN = LEN (ITEM, I)
      $CAP = CAP (ITEM, I)
      $BCAP= BCAP (ITEM, I)
```

TABLE VI-17 (Continued)

KEND FORTRAN

```
      COURSE = I + (ITEM * 1000)
C
C TEST FOR CONVENINGS. SKIP IF CONVENINGS LE 0
C
      IF ($CONV) 500, 500, 51
C
C
C SIMULTANEOUS COURSES SET UP FOR THOSE COURSES THAT MEET MORE THAN ONCE
C PER WEEK (50 CONV), OR FOR COURSES WHOSE LENGTH IS SUCH THAT NEW
C SESSIONS CONVENE BEFORE THE PREVIOUS SESSION GRADUATES.
C
C
C SET UP COUNTER 'SIMUL' TO INDICATE NUMBER OF SESSIONS OF THE SAME
C COURSE CONVENING SIMULTANEOUSLY DUE TO CONVENING FREQ GT THAN 1 / WEEK
C MULTIPLE CONVENING SCHEDULES SET UP AS NEEDED IN ARRAY C (I), MAX 10.
C
   51 DIVSR = (($CONV - 1) /50) + 1
      QUOT = $CONV ; SIMUL = DIVSR
C
C NOW BREAK UP COURSE INTO SEPARATE SECTIONS, WITH APPROX SAME NUMBER
C OF INDIVIDUAL CONVENINGS, AND EXACTLY SAME NUMBER TOTAL CONV.
C
      DO 100 I1 = 1, SIMUL
      IF (DIVSR) 100, 100, 99
   99 C(I1) = QUOT / DIVSR
      DIVSR = DIVSR - 1
      QUOT = QUOT - C (I1)
  100 CONTINUE
C
C CALC OFFSET - OFFSET AVAILABLE AS A MODEL INPUT TO DELAY OR ADVANCE
C A COURSE CONVENING SCHEDULE
C
C TEST FOR OFFSET
      IF (WKOFF) 152, 151, 152
C
C NO OFFSET, SET WEEKS & QUARTERS TO ACTUAL
C
  151 Q1 = Q ; W1 = W ; GO TO 160
C
C HAVE OFFSET, STRAIGHTEN OUT IF NEGATIVE (PRESTART)
  152 ITEM = WKOFF
C
C MODIFY QUARTERS & WEEKS TO WEEKS
      ITEM1 = ((Q - 1) * 13) + W
C
C SUBTRACT OFFSET
      ITEM1 = ITEM1 - ITEM
C
C SKIP CRSRLS IF 0 OR NEG
      IF (ITEM1) 500, 500, 153
C
C WEEKS POSITIVE, CLAMP TO 52
  153 IF (ITEM1 .GT. 52) ITEM1 = ITEM1 - 52
C
C CONVERT BACK INTO WORKING WEEK FOR SCHEDULE MATRIX
```

TABLE VI-17 (Continued)

KEND FORTRAN

```
C
      Q1 = (ITEM1 / 13) + 1
      W1 = ITEM1 - ((Q1 - 1) * 13)
C
  160 DO 200 I2 = 1, SIMUL
      ITEM = C (I2) ; IF (ITEM) 200, 200, 199
  199 ITEM1 = ISCHD (Q1,ITEM) / DIV (W1)
C
      IF ((( ITEM1/ 2) * 2) .EQ. ITEM1)  GO TO 200
C
C COURSE SECTION CONVENES THIS WEEK, CALL COURSE RELEASE -TO ACTIVE MTRX
C
      CALL CRSRLS (I)
C
  200 CONTINUE
  500 CONTINUE
C
C SAVE NUMBER OF COURSES RELEASED (OR ACTIVE) THIS WEEK
      NUMRUN = LAST
C
C
C
C **** END COURSE RELEASE - NOW CHECK ACTIVE MATRIX FOR COMPLETIONS ****
C
C
C SEARCH COURSE ACTIVE MATRIX FOR COURSES THAT END THIS WEEK
C ACTIVE (I) CONTAINS RELATIVE COURSE NUMBER, CRS ACTIVE IF GT ZERO
C
C DON'T TERMINATE OR COUNT DOWN COURSES DURING CHRISTMAS RECESS.
C
      IF (Q .EQ. 2 .AND. W .EQ. 13) GO TO 601
      IF (Q .EQ. 3 .AND. W .EQ.  1) GO TO 601
C
C SKIP COURSE END IF NEGFLAG (PRESTART) IS ON
      IF (NEGFLG .GT. 0) GO TO 601
C
      DO 600 I = 1, 250
      IF (ACTIVE (I)) 600, 600, 501
C
C COURSE ACTIVE - SUBTRACT 1 FROM WEEKS TO END & TEST IF COMPLETE
C ACTCTR (I) CONTAINS WEEKS TILL CRS END, COURSE ENDS IF 0 OR NEGATIVE
C
  501 NUMRUN = I
      ACTCTR (I) = ACTCTR (I) - 1.
      IF (ACTCTR (I)) 502, 502, 600
C
C COURSE ENDS THIS WEEK - CALL COURSE END SUBROUTINE
C
  502 CALL CRSEND (I)
C
  600 CONTINUE
  601 CONTINUE
C
C////////////////////////////////////////////////////////////////////////
C TRACE PRINT ROUTINE - PRINT ONLY IF PROPER TRACE FLAG IS SET
```

TABLE VI-17 (Continued)

KEND FORTRAN

```
      IF (TRACEF) 800, 800, 10602
10602 WRITE (6, 11134) W, Q, YEAR, NUMRUN
11134 FORMAT (' TRACE- END OF WEEK    WEEK =', I3, ';   QUARTER =', I2,
     1';   YEAR =', I5, 5X, 'COURSES ACTIVE THIS WEEK =', I4,/)
C////////////////////////////////////////////////////////////////////////
C ****** END WEEKLY COURSE LOOP - END COURSE START AND COMPLETE ******
C
  800 CONTINUE
C
C TEST IF NEGFLG IS ON, INDICATING THAT QUARTER RUN PRIOR TO THE START
C OF MODEL FISCAL YEAR CYCLING.
C
      IF (NEGFLG) 802, 802, 801
C
C NEGFLG ON, CALL SUBROUTINE PRERUN, CLEAR FLAG, AND BRANCH TO BEGINNING
C OF QUARTER CYCLING
C
  801 CALL PRERUN (2)
      NEGFLG = 0 ; GO TO 10800
  802 CONTINUE
C
C END QUARTER LOOP - CALL QRTEND SUBROUTINE TO COLLECT QUARTERLY DATA.
C
      CALL QRTEND (Q)
C
  900 CONTINUE
C
C END ANNUAL (4 QUARTERS) LOOP - CALL YEAR END SUBROUTINE TO AGGREGATE
C AND PRINT ANNUAL TOTALS.
C
      CALL YEARND (Y)
C
  999 CONTINUE
C
 1998 WRITE (6, 5000) CRSES
      IF (YRFLG .LE. 1) WRITE (11, 5000) CRSES
 5000 FORMAT ('1END OF TPF OUTPUT', I4, ' COURSES PROCESSED', / '/'/'*')
C
      STOP
      END

      SUBROUTINE PRERUN (X)
C
      IMPLICIT INTEGER*2 (A-Z,$)
C
C COMMON DATA FOR CLASS DATA MATRIX
C
      REAL*4 LEN, NOSHOW, NOSHPL, BKLGPL, BPOOL, NADIS, FAIL, CDP
      REAL*4 TDMAND, SETBAK, DATE
C
      COMMON /DATA/ CDP (328), CRSRUN (328), CONV (3,328), LEN (3,328)
      COMMON /DATA/ CAP (3,328), BCAP (3,328), CHGWK (2,328)
      COMMON /DATA/ OFFSET (328), DEMAND (3,328), BDMAND (3,328)
      COMMON /DATA/ BPOOL (328), FAIL (328), NADIS (328), NOSHPL (328)
      COMMON /DATA/ BAKLOG (328), NOSHOW (328), BKLGPL (328)
```

TABLE VI-17 (Continued)

```
C BUPERS HAS DEMAND - CALC DEMAND PER CONVENING.
C TEM EQUALS BUPERS STUDENTS (REAL).
C
  101 TEM = 0.
      TEM1 = $CONV
      IF ($CONV) 103, 103, 102
  102 TEM = $BDMND / TEM1
C
C NOW PUT ANY FRACTIONAL DEMAND INTO SAVE POOL. ITEM NOW EQ TO BUPERS
C STUDENTS (INTEGER).
C
  103 ITEM = TEM
      BPOOL (X) = BPOOL (X) + (TEM - ITEM)
C
      IF (BPOOL (X) .LT. 1) GO TO 105
C
C POOL HAS MORE THAN 1 STUDENT, RELEASE & CLEAR POOL OF WHOLE STUDENTS.
C
      ITEM1 = BPOOL (X)
      BPOOL (X) = BPOOL (X) - ITEM1
      ITEM = ITEM + ITEM1
C
C END BUPERS DEMAND. ITEM = NUMBER OF STUDENTS. 0 IF NO BUREAU DEMAND.
C ENTRY IF NO BP DEMAND.
C
C NOW RELEASE LOCAL QUOTA CONTROL DEMAND - SEATS AVAILABLE = CAPACITY
C MINUS BUREAU CONTROLLED SEATS.
C
  105 ITEM1 = $CAP - $BCAP
      IF (ITEM1) 11105, 11106, 11106
11105 ITEM1 = 0
C
C FIND LOCAL DEMAND FOR COURSE - DEMAND = ANNUAL DEMAND / CONVENINGS.
C ADD THIS DEMAND (TEM) TO BKLGPL OR USE AS DEMAND IF IN PRERUN
C IF IN INIT (PRERUN) MODE-SET UP CLASSES BUT DONT ADJUST POOLS
C
11106 TEM = $DMAND / $CONV
      IF (NEGFLG) 107, 107, 106
C
C HERE IF NEG FLG ON- HAVE BUPERS(ITEM), NOW ADD LOCAL - NO SHOW (TEM)
  106 ITEM2 = TEM - (TEM * NOSHOW (X))
      IF (ITEM2 .GT. ITEM1) ITEM2 = ITEM1
      ACTUTL (Y) = ITEM + ITEM2
      GO TO 999
C
C HERE IF NORMAL RELEASE - ADJUST POOLS ETC
C
  107 BKLGPL (X) = BKLGPL (X) + TEM
C
C RELEASE LOCAL STUDENTS - MAKE INIT ASSUMPTION ENTIRE POOL CAN FIT.
C ITEM2 EQUALS LOCAL QUOTA CONTROL STUDENTS.
C
      ITEM2 = BKLGPL (X)
C
C TEST IF POOL GT CAPACITY.
```

TABLE VI-17 (Continued)

```
C
C EXIT SUBROUTINE
C
      GO TO 999
C
C HAVE FOUND OPENING IN TABLE. OPENING IS 'Y', COURSE POINTER IS 'X'.
C TEST IF NEGFLG IS ON ( INDICATES MODEL PRESTART CONVENING).
C
   50 IF (NEGFLG) 60, 60, 51
C
C HERE IF FLAG IS ON - CALL PRESTART SUBROUTINE TO TEST IF COURSE SHOULD
C BE STARTED.  INIT RETURN CODE (DUMMY) TO 0
C
   51 DUMMY = 0
      CALL PRERUN (0)
C
C RETURN CODE 1 IF COURSE TO BE STARTED, 0 IF TO SKIP (DUMMY)
C IF COURSE TO BE STARTED, SKIP OVER QUARTERLY TOTAL ACCUMULATE.
C
      IF (DUMMY) 999, 999, 63
C
C COURSE SHOULD BE RELEASED NORMALLY - ACCUMULATE TOTALS IN QUARTER MTRX
C FOR CONVENINGS AND CAPACITY.
C
   60 TCONV (Q, X) = TCONV (Q, X) + 1
      TCAP (Q, X) = TCAP (Q, X) + SCAP
C
C LOAD ACTIVE MATRIX & BUMP CRSRUN COUNTER IN COURSE MATRIX, PRESTART
C COURSES ENTER ROUTINE HERE.  SAVE LAST AS NEXT POINTER TO ACTIVE MATRX
C
   63 CRSRUN (X) = CRSRUN (X) + 1
      ACTIVE (Y) = COURSE
      ACTCTR (Y) = SLEN
      ACTNEG (Y) = 0
      LAST = Y
C
C ROUTINE TO ALLOW MULTI YEAR RUN WITH ONLY 1 YEAR SCHEDULE DATA.
C CRITERIA, USE REVISED DEMAND IF AVAILABLE, USE CHANGE DATA IF CNG FLG
C
      ITEM = COURSE / 1000
      IF (YEARUN .LT. 2) GO TO 70
      IF (ITEM .GT. 1) GO TO 70
      IF (DEMAND (YEARUN, X)) 70, 70, 65
   65 ITEM = YEARUN
   70 SDMAND = DEMAND (ITEM, X) -BDMAND (ITEM, X)
      IF (SDMAND) 71, 72, 72
   71 SDMAND = 0
   72 SBDMND = BDMAND (ITEM, X)
C
C CHECK IF BUPERS HAS ANY CONTROLLED DEMAND FOR THIS COURSE-ITEM IS NONE
C CURRENTLY MODEL ALLOWS BUPERS TO OVERBOOK THERE COURSE
C
      ITEM = 0
      IF (SBDMND) 105, 105, 101
C
```

TABLE VI-17 (Continued)

KEND FORTRAN

```
C
C COMMON DATA FOR CLASS DATA MATRIX
C
      REAL*4 LEN, NOSHOW, NOSHPL, BKLGPL, BPOOL, NADIS, FAIL, CDP
      REAL*4 TDMAND, SETBAK, DATE
C
      COMMON /DATA/ CDP (328), CRSRUN (328), CONV (3,328), LEN (3,328)
      COMMON /DATA/ CAP (3,328), BCAP (3,328), CHGWK (2,328)
      COMMON /DATA/ OFFSET (328), DEMAND (3,328), BDMAND (3,328)
      COMMON /DATA/ BPOOL (328), FAIL (328), NADIS (328), NOSHPL (328)
      COMMON /DATA/ BAKLOG (328), NOSHOW (328), BKLGPL (328)
      COMMON /DATA/ BKLG (328), TDMAND, SETBAK, DATE
C
C QUARTERLY AND ANNUAL COURSE TOTALS ACCUMULATION
C
      REAL*4 TFAIL, TNADIS
      INTEGER*4 TSDAYS, TCAP, TUTIL
C
      COMMON /TOTALS/ TCAP (5, 328),   TUTIL (5, 328),  TSETBK (5, 328)
      COMMON /TOTALS/ TSDAYS (5, 328), TFAIL (5, 328),  TNOSHO (5, 328)
      COMMON /TOTALS/ TNADIS (5, 328), TBKLOG (5, 328), TCONV (5, 328)
C
C COMMON FOR COURSE ACTIVE MATRIX ELEMENTS
C
      REAL*4 ACTCTR
      COMMON /RUN/ ACTIVE (250), ACTCTR (250), ACTUTL (250), ACTNEG(250)
C
C COMMON FOR COMMUNICATION BETWEEN SUBROUTINES
C
      INTEGER*4 ITEM, ITEM1, ITEM2, ITEM3, ITEM4, ITEM5, WHERE
      REAL*4 TEM, TEM1, TEM2, TEM3, TEM4, TEM5, $LEN
C
      COMMON DUMMY, $CONV, $LEN, $CAP, $BCAP, LAST, COURSE
      COMMON $R1,$R2,$R3,$R4, TEM, TEM1, TEM2, TEM3, TEM4, TEM5
      COMMON ITEM, ITEM1, ITEM2, ITEM3, ITEM4, ITEM5
      COMMON YEAR, CRSES, Q, W, WKOFF, INIT, WHERE, NEGFLG, AOBCON
      COMMON TRACEF, RLSFLG, ENDFLG, PREFLG, YRFLG,YEARUN,PRINTF,NGR
C
C
C SEARCH ACTIVE MATRIX FOR FIRST AVAILABLE SLOT FOR NEW ACTIVE CONVNING.
C ACTIVE (Y) IS ZERO IF NOT IN USE, HAS POINTER TO CRSE MATRIX IF IN USE
C LAST IS POSITION LAST USED TO INSERT A COURSE, START SEARCH FROM THERE
C LAST IS 1 IF THIS IS FIRST COURSE RELEASE FOR THE WEEK.
C
C TEST IF LOCATION UNUSED IN ACTIVE MATRIX, EXIT IF FREE.
C
      DO 10 Y = LAST, 250
      IF (ACTIVE (Y)) 50, 50, 10
   10 CONTINUE
C
C HERE IF TABLE FULL - CANNOT RELEASE COURSE-WRITE ERROR MESSAGE.
C
      WRITE (6, 1000)  (CDP (X))
 1000 FORMAT (1H0, 'COURSE ACTIVE TABLE FULL - COURSE NOT RELEASED',
     1 '   COURSE NUMBER IS', A6)
```

TABLE VI-17 (Continued)

KEND FORTRAN

```
      IF (ACTIVE (J)) 500, 500, 401
C
C ACTIVE COURSE AT 'J'. CALC POSITION IN CLASS DATA MATRIX
C
  401 ITEM = ACTIVE (J) / 1000
      ITEM = ACTIVE (J) - ITEM * 1000
C
C FIND LENGTH FOR NEGDAYS CALCULATION.
C ACTCTR CONTAINS NUMBER OF WEEKS REMAINING TILL CLOSEOUT OF COURSE.
C
      $LEN = LEN (1,ITEM) - ACTCTR (J)
C
C//////////////////////////////////////////////////////////////////////////
C TRACE PRINT ROUTINE - PRINT ONLY IF PROPER TRACE FLAG IS SET
      IF (PREFLG) 403, 403, 402
  402 WRITE (6, 4356) CDP (ITEM), ACTCTR (J), $LEN
 4356 FORMAT (' PRESTART CLOSEOUT  CDP =', A4, 3X, 'ACTCTR = ', F5.1,
     1 '   NEGLEN = ', F5.1)
C//////////////////////////////////////////////////////////////////////////
C
C TEST AOB CONSTANT FOR WHICH METHOD OF CALC STUDENT DAYS
  403 IF (AOBCON .GT. 300) GO TO 450
C
C HERE IF CLASS DAYS METHOD USED TO CALCULATE AOB.
C
      ACTNEG (J) = $LEN * ACTUTL (J)  *5.    + .01
      GO TO 500
C
C HERE IF CALANDER DAYS METHOD USED FOR AOB CALCULATION.
C
  450 ITEM1 = $LEN - 1
      TEM = (ITEM1 * 2) + ($LEN * 5)
      ACTNEG (J) = TEM * ACTUTL (J) + .01
C
  500 CONTINUE
C HERE IF NO MORE ACTIVE COURSES - EXIT SUBROUTINE
C
      NEGFLG = 0
C
 1999 CONTINUE
C
      RETURN
      END

      SUBROUTINE CRSRLS (X)
C
C COURSE RELEASE (START) SUBROUTINE - RELEASES COURSE, CALCULATES BUPERS
C SEATS & ATTENDANCE, LOCAL QUOTA CONTROL SCHEDULING, BACKLOG, NOSHOWS,
C SUBSTITUTE QUOTAS, AND ULTIMATE CLASS UTILIZATION, INCLUDING SETBACKS.
C
C COURSE TOTALS - ADDED TO QUARTER TOTALS AS COURSE RELEASES:
C CAPACITY, UTILIZATION, NOSHOWS, BACKLOG, SUBSTITUTE QUOTAS.
C
      IMPLICIT INTEGER*2 (A-Z,$)
      REAL*4 SUMAND, PADAND
```

TABLE VI-17 (Continued)

KEND FORTRAN

```
      COMMON /DATA/ BKLG (328), TOMAND, SETBAK, DATE
C
C COMMON FOR COURSE ACTIVE MATRIX ELEMENTS
C
      REAL*4 ACTCTR
      COMMON /RUN/ ACTIVE (250), ACTCTR (250), ACTUTL (250), ACTNEG(250)
C
C COMMON FOR COMMUNICATION BETWEEN SUBROUTINES
C
      INTEGER*4 ITEM, ITEM1, ITEM2, ITEM3, ITEM4, ITEM5, WHERE
      REAL*4 TEM, TEM1, TEM2, TEM3, TEM4, TEM5, $LEN
C
      COMMON DUMMY, $CONV, $LEN, $CAP, $BCAP, LAST, COURSE
      COMMON $R1,$R2,$R3,$R4, TEM, TEM1, TEM2, TEM3, TEM4, TEM5
      COMMON ITEM, ITEM1, ITEM2, ITEM3, ITEM4, ITEM5
      COMMON YEAR, CRSES, Q, W, WKOFF, INIT, WHERE, NEGFLG, AOBCON
      COMMON TRACEF, RLSFLG, ENDFLG, PREFLG, YRFLG,YEARUN,PRINTF,NGR
C
C
C THIS SUBROUTINE HANDLES THE UNIQUE SITUATION WHERE A NEGATIVE OFFSET
C HAS BEEN SPECIFIED TO START A COURSE PRIOR TO THE START OF A MODEL RUN
C THIS SITUATION EXISTS FOR COURSES OF RELATIVELY LONG DURATION, AND
C RELATIVELY FEW CONVENINGS. IT IS USEFUL TO INDICATE TRUE AOB DURING
C A QUARTER.
C
C A CODE OF 0 IS IN (X) IF CALLED BY COURSE RELEASE, 2 IF CALLED BY END.
C
      IF (X) 100, 100, 1000
C
C CALLED BY RELEASE, TEST IF COURSE SHOULD BE RELEASED. CRITERIA IS THAT
C COURSE LENGTH IS SUCH THAT COURSE WILL STILL BE RUNNING AT MODEL START
C TEST INIT TO SEE IF GENERAL PRERUN OR NEG OFFSET PRERUN
C
C
  100 TEM = 14. - W
      IF ($LEN .LE. TEM) GO TO 1999
C
C COURSE SHOULD BE RELEASED - SET DUMMY TO 1 FOR RETURN.
C MODIFY LENGTH TO REFLECT WEEKS REMAINING IN 1ST QUARTER OF COURSE.
C
      DUMMY = 1 ; $LEN = $LEN - TEM
C
C//////////////////////////////////////////////////////////////////////////
C TRACE PRINT ROUTINE - PRINT ONLY IF PROPER TRACE FLAG IS SET
      IF (PREFLG) 1999, 1999, 3001
 3001  WRITE (6, 3000) CDP (COURSE), COURSE, WKOFF, $LEN
 3000 FORMAT (' PRERUN CALLED - CDP = ', A4, '  POSITION =', I5,
     1 '  OFFSET =', I4, '  LENGTH IN 1ST QTR =', F4.1)
      GO TO 1999
C//////////////////////////////////////////////////////////////////////////
C
C
C HERE IF PRERUN CLOSEOUT ROUTINE
C
 1000 DO 500 J = 1, 250
```

TABLE VI-17 (Continued)

KEND FORTRAN

```
C CLAMP TO CAPACITY IF GREATER.
C ITEM1 IS NOW LOCAL QUOTA CONTROL CAPACITY
C
      IF (ITEM2 .GT. ITEM1) ITEM2 = ITEM1
C
C RELEASE THESE STUDENTS TO COURSE. CLEAR FROM POOL.
      BKLGPL (X) = BKLGPL (X) - ITEM2
C
C NOW TEST FOR LOCAL NO SHOWS
      NOSHPL (X) = NOSHPL (X) + (ITEM2 * NOSHOW (X))
C
C SEE IF GT 1 IN POOL.
      ITEM3 = NOSHPL (X)
      IF (ITEM3) 111, 111, 110
C
C NO SHOWS GT 1, ADD TO TOTALS & TAKE FROM CLASS.
C ITEM3 IS STUDENTS FROM NO SHOW POOL
C
  110 TNOSHO (Q, X) = TNOSHO (Q, X) + ITEM3
      ITEM2 = ITEM2 - ITEM3
      NOSHPL (X) = NOSHPL (X) - ITEM3
  111 CONTINUE
C
C NOW TOTAL STUDENTS & SEE IF FULL CAPACITY.
C ITEM IS BUPERS STUDENTS, ITEM2 IS LOCAL STUDENTS AFTER NO SHOW
C
      ACTUTL (Y) = ITEM + ITEM2
C
      ITEM4 = 0
      IF (ACTUTL (Y) .GE. $CAP) GO TO 114
      ITEM4 = (.1 * $CAP) + .3
C
C CAPACITY NOT FULL - CHECK IF CAN RELEASE THESE FROM BACKLOG POOL
C
      ITEM5 = BKLGPL (X)
      IF (ITEM5) 114, 114, 113
C
C BACKLOG POOL GT 1, RELEASE SUBSTITUTE QUOTAS
C CLAMP STUDENTS TO MAX IN POOL.
C
  113 IF (ITEM5 .LT. ITEM4) ITEM4 = ITEM5
      ACTUTL (Y) = ACTUTL (Y) + ITEM4
C
  114 TUTIL (Q,X) = TUTIL (Q,X) + ACTUTL (Y)
  999 CONTINUE
C
C////////////////////////////////////////////////////////////////////////
C TRACE PRINT ROUTINE - PRINT ONLY IF PROPER TRACE FLAG IS SET
      IF (RLSFLG) 1010, 1010, 1001
 1001 WRITE (6, 1002) CDP (X), ACTUTL (Y), $LEN, X, LAST
     1 , ITEM, ITEM2, ITEM4, ITEM3
 1002 FORMAT (' RELEASE- CDP= ', A4, '; STUDENTS= ', I3, '; LENGTH='
     1,F5.1,'; CRS SEQ=',I4,'; RUN POSITN=', I4, '; BUPRS=', I3, '
     2LOCAL=', I3, '; SUBQTA=', I2,'; NOSHO=', I3)
C
```

TABLE VI-17 (Continued)

KEND FORTRAN

```
      IF (NEGFLG .GT. 0) WRITE (6, 1003) DUMMY
 1003 FORMAT ('+', T132, 'P', I1)
 1010 CONTINUE
C////////////////////////////////////////////////////////////////////////
      RETURN
      END
      SUBROUTINE CRSEND (X)
C SUBROUTINE CRSEND CALCULATES PASS/FAIL/SETBACK, NACDISENROLLMENT,
C AND CALCULATES UTILIZATION, TOTALS FOR QUARTERLY REPORT
C
      IMPLICIT INTEGER*2 (A-Z,$)
C COMMON DATA FOR CLASS DATA MATRIX
      REAL*4 LEN, NOSHOW, NOSHPL, BKLGPL, BPOOL, NADIS, FAIL, CDP
      REAL*4 TDMAND, SETBAK, DATE
      COMMON /DATA/ CDP (328), CRSRUN (328), CONV (3,328), LEN (3,328)
      COMMON /DATA/ CAP (3,328), RCAP (3,328), CHGWK (2,328)
      COMMON /DATA/ OFFSET (328), DEMAND (3,328), BDMAND (3,328)
      COMMON /DATA/ BPOOL (328), FAIL (328), NADIS (328), NOSHPL (328)
      COMMON /DATA/ BAKLOG (328), NOSHOW (328), BKLGPL (328)
      COMMON /DATA/ BKLG (328), TDMAND, SETBAK, DATE
C
C QUARTERLY AND ANNUAL COURSE TOTALS ACCUMULATION
      REAL*4 TFAIL, TNADIS
      INTEGER*4 TSDAYS, TCAP, TUTIL
      COMMON /TOTALS/ TCAP (5, 328),   TUTIL (5, 328),  TSETBK (5, 328)
      COMMON /TOTALS/ TSDAYS (5, 328), TFAIL (5, 328),  TNOSHO (5, 328)
      COMMON /TOTALS/ TNADIS (5, 328), TBKLOG (5, 328), TCONV (5, 328)
C
C COMMON FOR COURSE ACTIVE MATRIX ELEMENTS
      REAL*4 ACTCTR
      COMMON /RUN/ ACTIVE (250), ACTCTR (250), ACTUTL (250), ACTNEG(250)
C
C COMMON FOR COMMUNICATION BETWEEN SUBROUTINES
      INTEGER*4 ITEM, ITEM1, ITEM2, ITEM3, ITEM4, ITEM5, WHERE
      REAL*4 TEM, TEM1, TEM2, TEM3, TEM4, TEM5, $LEN
      COMMON DUMMY, $CONV, $LEN, $CAP, $BCAP, LAST, COURSE
      COMMON $R1,$R2,$R3,$R4, TEM, TEM1, TEM2, TEM3, TEM4, TEM5
      COMMON ITEM, ITEM1, ITEM2, ITEM3, ITEM4, ITEM5
      COMMON YEAR, CRSES, G, W, WKOFF, INIT, WHERE, NEGFLG, AOBCON
      COMMON TRACEF, RLSFLG, ENDFLG, PREFLG, YRFLG,YEARUN,PRINTF,NGR
C
      I2 =  0; ITEM2 = 0 ; J = ACTIVE (X)
C
C CALLING PARAMETER 'X' IS SUBSCRIPT OF COURSE IN ACTIVE MATRIX.
C ACTIVE (X) CONTAINS RELATIVE COURSE NUMBER IN COURSE MATRIX, HAS 1000
C ADDED IF '1' SCHED, 2000 IF '2' SCHED, 3000 IF '3' SCHED
C
      ITEM5 = J / 1000
      J = J - ITEM5 * 1000
C
      $LEN = LEN (ITEM5, J)
C
C CALCULATE FAILURES - FAILURE RATE FROM REGRESSION CONSTANTS
C NOTE - J POINTS TO COURSE MATRIX, X POINTS TO ACTIVE MATRIX
C
```

KEND FORTRAN

```
      TEM = FAIL (J) * ACTUTL (X)
      TFAIL (Q, J) = TFAIL (Q, J) + TEM
C
C CALCULATE NON-ACADEMIC DISENROLLMENT - RATE FROM DATA BASE
C
      TEM1 = NADIS (J) * ACTUTL (X)
      TNADIS (Q, J) = TNADIS (Q, J) + TEM1
C
C TEST FOR SETBACK POSSIBILITY
C CRITERIA FOR SETBACK: COURSE GT 2 WEEKS, MULTIPLE CONVENINGS, FAILURE
C RATE TO PRODUCE SETBACKS.
C
      IF ($LEN .LT. 2.0) GO TO 100
      IF (CRSRUN (J) .LT. 2) GO TO 100
      TEM2 = TEM * SETBAK + .5
      IF (TEM2 .LT. 1.) GO TO 100
C
C HERE IF SETBACKS MAY OCCUR
C
      GO TO 200
C
C TOTAL COURSE AND CLOSEOUT ACTIVE MATRIX FOR COURSE
C
  100 ACTIVE (X) = 0
C
C CALCULATE TOTAL STUDENT DAYS FOR AOB CALCULATION.
C ACTNEG CONTAINS DAYS CHARGED TO OTHER QUARTER OR OTHER CONVENINGS.
C
C TEST WHICH METHOD OF AOB CALCULATION TO BE USED.
      IF (AOBCON .GT. 300) GO TO 430
C
C HERE IF CLASS DAYS METHOD TO BE USED
      TSDAYS (Q, J) = TSDAYS (Q, J) + ((($LEN * 5. * ACTUTL (X))
     1 + .01) - ACTNEG (X)
      GO TO 440
C
C HERE IF YEARLY METHOD TO BE USED.
C
  430 ITEM1 = $LEN - 1
      TEM = (ITEM1 * 2) + ($LEN * 5.)
      TSDAYS (Q, J) = TSDAYS (Q, J) +(TEM * ACTUTL (X) +.01)
     1 - ACTNEG (X)
  440 CONTINUE
C
C CLOSE OUT COURSE, RESET ACTIVE & RUN FLAGS.
C
      ACTIVE (X) = 0
      CRSRUN (J) = CRSRUN (J) - 1
      IF (CRSRUN (J) .LT. 0) CRSRUN (J) = 0
      GO TO 999
C ********** SETBACK ROUTINE **********
C HAVE SETBACKS - SEARCH OUT PROPER CONVENING TO SET BACK TO.
C
C INIT TEST VALUES
  200 ITEM5 = 1
```

TABLE VI-17 (Continued)

KEND FORTRAN

```
      ITEM = ACTIVE (X)
      TEM5 = 999. ; ITEM2 = 0 ; ITEM3 = 0
      DO 120 I2 = 1. 250
      ITEM1 = ACTIVE (I2)
      IF (ITEM1) 120. 120. 101
C HAVE ACTIVE COURSE - TEST IF OURS: DUMMY=SCHED. ITEM1=CRSNO
  101 DUMMY = ITEM1 / 1000
      ITEM1 = ITEM1 - DUMMY * 1000
      IF (ITEM1 .NE. J) GO TO 120
C HAVE OUR COURSE NUMBER-TEST IF ONE WE ARE ENDING
      IF (I2 .EQ. X) GO TO 118
C FOUND ONE OF OUR COURSES-TEST IF EARLIEST CONVENING.
      IF (ACTCTR (I2) .GT. TEM5) GO TO 118
C EARLIEST SO FAR. SAVE WEEKS & POSITION
      TEM5 = ACTCTR (I2) ; ITEM3 = I2
C SEE IF WE FOUND ALL CONVENINGS. BUMP COUNTER & TEST.
  119 ITEM2 = ITEM2 + 1
      IF (ITEM2 .EQ. CRSRUN (J)) GO TO 121
  120 CONTINUE
C
C HERE IF DONE COURSE SEARCH OF ACTIVE MATRIX. ITEM3 IS 0 IF NO HIT.
C STATEMENT 100 IS BACKWARD BRANCH TO CLOSEOUT ROUTINE
C
  121 IF (ITEM3) 100. 100. 122
C
C FIND COURSE AND MODIFY TABLES TO REPRESENT SETBACKS
C
  122 ITEM = ACTIVE (ITEM3)
      DUMMY = ITEM / 1000
      ITEM = ITEM - DUMMY * 1000
      TEM4 = LEN (DUMMY. ITEM)
C
      ITEM2 = TEM2
      ACTUTL (ITEM3) = ACTUTL (ITEM3) + ITEM2
      TSETBK (Q. ITEM) = TSETBK (Q. ITEM) + ITEM2
C NOW CALCULATE NEGATIVE STUDENT DAYS - TO COMPENSATE COURSE STUD LEFT.
C ITEM2 IS SETBACKS. LEN IS FIRST COURSE LENGTH. TEM4 IS SET BACK TO CRS
C LENGTH. TEM5 IS WKS LEFT IN SET BACK TO COURSE.
C
C NOW LET TEM5 EQUAL WEEKS STUDENT HAS BEEN IN SET BACK TO COURSE.
      TEM5 = TEM4 - TEM5
C
C TEST AOB MULT FOR METHOD OF AOB CALCULATION.
      IF (AOBCON .GT. 300) GO TO 130
C
C AOBCON LT 300. USE STUDENT DAYS METHOD.
      ACTNEG (X) = ACTNEG (X) +((TEM5 * 5. * TEM2) + .01)
      GO TO 100
C
C YEARLY AOB METHOD USED - CONVERT WEEKS TO INTEGER.
C STUDENT BEGINS SETBACK TO CLASS MONDAY. WEEKEND CHARGED TO OLD COURSE
C CALC TOTAL STUD DAYS. INCLUDING WEEKENDS - DUMMY IS INTEGER.
C
  130 DUMMY = TEM5
      TEM5 = (DUMMY * 2) +(TEM5 * 3.)
```

TABLE VI-17 (Continued)

KEND FORTRAN

```
      ACTNEG (X) = ACTNEG (X) +(TEM5 * TEM2 + .01)
      GO TO 100
  999 CONTINUE
C
C////////////////////////////////////////////////////////////////////////
      IF (ENDFLG) 1010, 1010, 1001
 1001 WRITE (6, 1002) CDP (J), ACTUTL (X),$LEN, J, X
 1002 FORMAT (' CRS END- CDP= ', A4, '; STUDENTS= ', I3, '; LENGTH='
     1,F5.1,'; CRS SEQ=',I4,'; RUN POSITN=', I4)
      IF (ITEM2 .GT. 0) WRITE (6, 1005) ITEM2, ITEM3
 1005 FORMAT ('+', T105, I2, ' STDNTS SET TO CRS ', I4)
 1010 CONTINUE
C////////////////////////////////////////////////////////////////////////
      RETURN
      END
      SUBROUTINE QRTEND (X)
C QUARTER END ROUTINE TOTALS STUDENT DAYS OF ALL CONVENINGS IN SESSION,
C AND ADDS THEM TO COURSE TOTALS.  THESE DAYS ARE ALSO ENTERED AS
C 'NEGATIVE' DAYS IN ACTIVE MATRIX TO PREVENT THEM FROM BEING COUNTED IN
C  NEXT QUARTERS TOTALS.
C ROUTINE CHECKS AOBCON FOR WHICH METHOD OF TOTALLING AOB IS USED.
C
      IMPLICIT INTEGER*2 (A-Z,$)
C COMMON DATA FOR CLASS DATA MATRIX
      REAL*4 LEN, NOSHOW, NOSHPL, BKLGPL, BPOOL, NADIS, FAIL, CDP
      REAL*4 TDMAND, SETBAK, DATE
      COMMON /DATA/ CDP (328), CRSRUN (328), CONV (3,328), LEN (3,328)
      COMMON /DATA/ CAP (3,328), BCAP (3,328), CHGWK (2,328)
      COMMON /DATA/ OFFSET (328), DEMAND (3,328), BDMAND (3,328)
      COMMON /DATA/ BPOOL (328), FAIL (328), NADIS (328), NOSHPL (328)
      COMMON /DATA/ BKLOG (328), NOSHOW (328), BKLGPL (328)
      COMMON /DATA/ BKLG (328), TDMAND, SETBAK, DATE
C
C QUARTERLY AND ANNUAL COURSE TOTALS ACCUMULATION
      REAL*4 TFAIL, TNADIS
      INTEGER*4 TSDAYS, TCAP, TUTIL
      COMMON /TOTALS/ TCAP (5, 328),   TUTIL (5, 328),  TSETBK (5, 328)
      COMMON /TOTALS/ TSDAYS (5, 328), TFAIL (5, 328),  TNOSHO (5, 328)
      COMMON /TOTALS/ TNADIS (5, 328), TBKLOG (5, 328), TCONV (5, 328)
C
C COMMON FOR COURSE ACTIVE MATRIX ELEMENTS
      REAL*4 ACTCTR
      COMMON /RUN/ ACTIVE (250), ACTCTR (250), ACTUTL (250), ACTNEG(250)
C COMMON FOR COMMUNICATION BETWEEN SUBROUTINES
      INTEGER*4 ITEM, ITEM1, ITEM2, ITEM3, ITEM4, ITEM5, WHERE
      REAL*4 TEM, TEM1, TEM2, TEM3, TEM4, TEM5, $LEN
      COMMON DUMMY, $CONV, $LEN, $CAP, $BCAP, LAST, COURSE
      COMMON $R1,$R2,$R3,$R4, TEM, TEM1, TEM2, TEM3, TEM4, TEM5
      COMMON ITEM, ITEM1, ITEM2, ITEM3, ITEM4, ITEM5
      COMMON YEAR, CRSES, Q, W, WKOFF, INIT, WHERE, NEGFLG, AOBCON
      COMMON TRACEF, RLSFLG, ENDFLG, PREFLG, YRFLG,YEARUN,PRINTF,NGR
C TEST ACTIVE MATRIX FOR ACTIVE COURSES.
      DO 500 J = 1, 250
      IF (ACTIVE (J)) 500, 500, 401
C
```

TABLE VI-17 (Continued)

```
C ACTIVE FILE HAS A COURSE ACTIVE AT 'J'.
C GET POINTER INTO COURSE MATRIX.  .
C
  401 ITEM = ACTIVE (J)
      DUMMY = ITEM / 1000
      ITEM = ITEM - DUMMY * 1000
      $LEN = LEN (DUMMY, ITEM)
C
C TEST WHICH METHOD OF AOH CALCULATION TO BE USED.
      IF (AOBCON .GT. 300) GO TO 430
C
C C HERE IF CLASS DAYS METHOD TO BE USED.
      TEM = ($LEN - ACTCTR (J)) * ACTUTL (J)      *5.
      GO TO 440
C
C HERE IF YEARLY AOH METHOD TO BE USED
C CHARGE THIS WEEKEND TO THE COURSE BEING CLOSED OUT
C
  430 TEM1 =$LEN - ACTCTR (J)
      ITEM1 = TEM1
      TEM = (ITEM1 * 2) + (TEM1 * 5)
      TEM = TEM * ACTUTL (J)
  440 CONTINUE
C
C ADD TOTAL STUDENT DAYS TO COURSE & ACTIVE MATRICES
C
      TEM = TEM + .01
      ACTNEG (J) = ACTNEG (J) + TEM
      TSDAYS (Q, ITEM) = TSDAYS (Q, ITEM) + TEM
  500 CONTINUE
C COPY CURRENT BACKLOG POOL FIGURES INTO COURSE TOTALS
      DO 8111 I = 1, CRSES
 8111 TBKLOG (Q,I) = BKLGPL (I)
C
C COPY FRACTIONAL FAIL & NADIS INTO NEXT QUARTER TOTALS, SKIP IF 4TH QTR
      IF (Q .GE. 4) GO TO 8113
      J = Q + 1
      DO 8112 I = 1, CRSES
      ITEM = TFAIL (Q,I)
      TFAIL (J,I) = TFAIL (Q,I) - ITEM
      ITEM = TNADIS (Q,I)
 8112 TNADIS (J,I) = TNADIS (Q,I) - ITEM
 8113 CONTINUE
C////////////////////////////////////////////////////////////////////////
      IF (TRACEF .GT. 0) WRITE (6, 1000) X
 1000 FORMAT (' QUARTER END ROUTINE FOR QUARTER ', I1)
C////////////////////////////////////////////////////////////////////////
      RETURN
      END
      SUBROUTINE YEARND (Y)
C THIS SUBROUTINE CALCULATES ANNUAL TOTALS AND CALLS THE PROPER SUBROUT-
C INE FOR PRINTOUT OR CRT OUTPUT.  IF A TRACE FLAG IS  ON  THIS SUBR IS
C BYPASSED AND A TRACE ONLY SUBROUTINE IS CALLED.
C TOTALS ARE AGGREGATED BY COURSE, SCHOOL, AND CENTER ON AN ANNUAL BASIS
C WITH NO CARRY OVER TOTALS FROM YEAR TO YEAR.
```

TABLE VI-17 (Continued)

KEND FORTRAN

```
      IMPLICIT INTEGER*2 (A-Z,$)
C COMMON FOR CLASS DATA MATRIX
      REAL*4 LEN, NOSHOW, NOSHPL, BKLGPL, BPOOL, NADIS, FAIL, CDP
      REAL*4 TDMAND, SETBAK, DATE
      COMMON /DATA/ CDP (328), CRSRUN (328), CONV (3,328), LEN (3,328)
      COMMON /DATA/ CAP (3,328), BCAP (3,328), CHGWK (2,328)
      COMMON /DATA/ OFFSET (328), DEMAND (3,328), BDMAND (3,328)
      COMMON /DATA/ BPOOL (328), FAIL (328), NADIS (328), NOSHPL (328)
      COMMON /DATA/ BAKLOG (328), NOSHOW (328), BKLGPL (328)
      COMMON /DATA/ BKLG (328), TDMAND, SETBAK, DATE
      INTEGER*4 PNEC
      REAL*4 PAOB, PFAIL
      REAL*8 NAME1, NAME2, CATNO1
      COMMON /TEXT/ PFAIL(328), NAME1 (328), NAME2 (328), CATNO1 (328)
      COMMON /TEXT/  CATNO2 (328), PNEC (328), TYPE (328), PAOB (328)
      COMMON /TEXT/ PRTY(328),SCH1(328),SCH2(328),SCH3(328),SCH4(328)
      REAL*4 TFAIL, TNADIS
      INTEGER*4 TSDAYS, TCAP, TUTIL
      COMMON /TOTALS/ TCAP (5, 328),   TUTIL (5, 328),  TSETBK (5, 328)
      COMMON /TOTALS/ TSDAYS (5, 328), TFAIL (5, 328),  TNOSHO (5, 328)
      COMMON /TOTALS/ TNADIS (5, 328), TBKLOG (5, 328), TCONV (5, 328)
C COMMON FOR SCHOOL AND CENTER TOTALS
      INTEGER*4 SUTIL, CUTIL, SCAP, CCAP, SSDAYS, CSDAYS, IFAIL, INADIS
      INTEGER*4 PASS, SBKLOG, CBKLOG, SBKL,CBKL
      REAL*4 AOB, PCTUTL, PCTNSH, PCTPAS, PCTFL, PCTSB, PCTNDS, PLEN
      COMMON /T/ SUTIL (5), SCAP (5), SSDAYS (5), SSETBK (5), SNOSHO (5)
      COMMON /T/ CUTIL (5), CCAP (5), CSDAYS (5), CSETBK (5), CNOSHO (5)
      COMMON /T/SBKL(5),SBKLOG(5),SCONV (5), SFAIL (5), SNADIS (5)
      COMMON /T/ CCONV(5),CBKL(5),CBKLOG(5), CFAIL (5), CNADIS (5)
      COMMON /T/ PCTUTL (5), PCTNSH (5), PCTNDS (5), INADIS (5)
      COMMON /T/ AOB (5), PASS (5), PCTPAS (5), PCTFL (5), PCTSB (5)
      COMMON /T/ IFAIL (5), PLEN (3), BKL (5), SPLAT (4), YEAR2D
C COMMON FOR COMMUNICATION BETWEEN SUBROUTINES
      INTEGER*4 ITEM, ITEM1, ITEM2, ITEM3, ITEM4, ITEM5, WHERE
      REAL*4 TEM, TEM1, TEM2, TEM3, TEM4, TEM5, $LEN
      COMMON DUMMY, $CONV, $LEN, $CAP, $BCAP, LAST, COURSE
      COMMON $R1,$R2,$R3,$R4, TEM, TEM1, TEM2, TEM3, TEM4, TEM5
      COMMON ITEM, ITEM1, ITEM2, ITEM3, ITEM4, ITEM5
      COMMON YEAR, CRSES, W, W, WKOFF, INIT, WHERE, NEGFLG, AOBCON
      COMMON TRACEF, RLSFLG, ENDFLG, PREFLG, YRFLG,YEARUN,PRINTF,NGR
C
C YRFLG 0 FOR CRT OUTPUT, 1 FOR BOTH, 2 FOR PRINT ONLY, 9 FOR TRACE ONLY
      IF (YRFLG .GE. 9) GO TO 29999
C
C SET SPLAT TO KEY YEAR OF RUN
      IF (YEARUN .LE. 1) GO TO 81001
      ITEM = YEARUN - 1
      SPLAT (YEARUN) = SPLAT (ITEM)
      SPLAT (ITEM) = SPLAT (4)
81001 CONTINUE
C
C INIT & CLAMP RUN CONSTANTS
      ITEM5 = 0
      IF (AOBCON .LE. 0) AOBCON = 250
      YEAR2D = YEAR - 1900
```

TABLE VI-17 (Continued)

```
C
C INITIALIZE CENTER TOTALS
C
      DO 1 Q = 1, 5
      CUTIL (Q) = 0
      CCAP  (Q) = 0
      CSETHR (Q) = 0
      CSDAYS (Q) = 0
      CNOSHO (Q) = 0
      CCONV  (Q) = 0
      CFAIL  (Q) = 0
      CNADIS (Q) = 0
      CHKL (Q) = 0
      CHKLOG (Q) = 0
C
      SUTIL (Q) = 0
      SCAP  (Q) = 0
      SSETHR(Q) = 0
      SSDAYS(Q) = 0
      SNOSHO(Q) = 0
      SCONV (Q) = 0
      SFAIL (Q) = 0
      SNADIS(Q) = 0
      SHKL (Q) = 0
    1 SHKLOG (Q) = 0
C
C
C CALCULATE TOTALS & PRINT FOR EACH COURSE
      DO 4444 I = 1, CRSES
C
C TEST FOR SCHOOL CODE CHANGE
      IF (I .EQ. 1) GO TO 4 ;/* SKIP SUMMARY IF 1ST COURSE */
      IF ($R2 .EQ. 0) GO TO 4 ;/* SKIP- NO SCHL SUMMARY */
      Q = I - 1 ; /* LAST COURSE PROCESSED */
      IF ($R2 .EQ. 1) GO TO 3 ;/* NO SCH SEL - KEY ON 1ST SCHL CHAR */
      IF (SCH2 (I) .NE. SCH2 (Q)) GO TO 500  ;/* CHG- PRINT SUMMARY */
      IF (SCH3 (I) .NE. SCH3 (Q)) GO TO 500
      IF (SCH4 (I) .NE. SCH4 (Q)) GO TO 500
    3 IF (SCH1 (I) .NE. SCH1 (Q)) GO TO 500
C
C RETURN FROM PRINT SCHOOL TOTALS - NOW CALC TOTALS FOR CLASS
C ITEM5 IS COUNTER FOR COURSE ACTUALLY CALCULATING
C
    4 ITEM5 = ITEM5 + 1
C
C USE 5TH POSITION IN TOTALS MATRIX TO AGGREGATE TOTALS - INIT TO ZERO
      TUTIL (5, I) = 0
      TCAP (5, I) = 0
      TSETHR (5, I) = 0
      TSDAYS (5, I) = 0
      TNOSHO (5, I) = 0
      TCONV (5, I) = 0
C
C INIT INTEGER FAIL & NON-ACADEMIC DROPOUT TOTALS TO ZERO
C
```

TABLE VI-17 (Continued)

KEND FORTRAN

```
      IFAIL (5) = 0
      INADIS (5) = 0
C
C
C NOW TOTAL ANNUAL FIGURES
C
      DO 5 Q = 1, 4
C
      TUTIL(5,I) = TUTIL(5,I) + TUTIL(Q,I)
      TCAP(5,I) = TCAP(5,I) + TCAP(Q,I)
      TSETBK(5,I) = TSETBK(5,I) + TSETBK(Q,I)
      TSDAYS(5,I) = TSDAYS(5,I) + TSDAYS(Q,I)
      TNOSHO (5, I) = TNOSHO (5, I) + TNOSHO (Q, I)
      TCONV(5,I) = TCONV(5,I) + TCONV(Q,I)
C
C CONVERT TO INTEGER, ROUND, & TOTAL FAILURES & NON-ACADEMIC DISENROLLS
C
      IFAIL (Q) = 0
      IFAIL (Q) = TFAIL (Q, I) + .5
      IFAIL (5) = IFAIL (5) + IFAIL (Q)
C
      INADIS (Q) = 0
      INADIS (Q) = TNADIS (Q, I) + .5
      INADIS (5) = INADIS (5) + INADIS (Q)
C
C TOTAL DATA FOR INDIVIDUAL SCHOOL TOTALS
C
      SUTIL (Q) = SUTIL (Q) + TUTIL (Q, I)
      SCAP (Q) = SCAP (Q) + TCAP (Q, I)
      SSETBK (Q) = SSETBK (Q) + TSETBK (Q, I)
      SSDAYS (Q) = SSDAYS (Q) + TSDAYS (Q, I)
      SNOSHO (Q) = SNOSHO (Q) + TNOSHO (Q, I)
      SCONV (Q) = SCONV (Q) + TCONV (Q, I)
      SFAIL (Q) = SFAIL (Q) + IFAIL (Q)
      SNADIS (Q) = SNADIS (Q) + TNADIS (Q, I)
C
C TOTAL DATA FOR FLEET TRAINING CENTER TOTALS
C
      CUTIL (Q) = CUTIL (Q) + TUTIL (Q, I)
      CCAP (Q) = CCAP (Q) + TCAP (Q, I)
      CSETBK (Q) = CSETBK (Q) + TSETBK (Q, I)
      CSDAYS (Q) = CSDAYS (Q) + TSDAYS (Q, I)
      CNOSHO (Q) = CNOSHO (Q) + TNOSHO (Q, I)
      CCONV (Q) = CCONV (Q) + TCONV (Q, I)
      CFAIL (Q) = CFAIL (Q) + IFAIL (Q)
      CNADIS (Q) = CNADIS (Q) + TNADIS (Q, I)
    5 CONTINUE
C
C NOW CALCULATE TOTALS & PERCENTAGES
C ASSUME CHANGE WEEK HAS NOT PASSED THIS QUARTER, INIT CAP & CONV TO 'A'
      SCONV = CONV (1,I)
      SCAP = CAP (1,I) - RCAP (1,I)
      TBKLOG (5, I) = TBKLOG (4, I)
C
C NOW CALCULATE UTILIZATIONS & PERCENTAGES
```

KEND FORTRAN

```
      DO 12 Q = 1, 5
      BKL (Q) = 0
      IF (TBKLOG (Q,I) .EQ. 0) GO TO 12
C
C TEST IF THERE IS A CHANGE WEEK
      IF (CHGWK (1,I)) 7, 7, 6
C
C HERE IF THERE IS A CHANGE WEEK, CHECK IF IT HAS PASSED
    6 ITEM = Q
      IF (ITEM .GT. 4) ITEM = 4
      ITEM = (13 * ITEM) + (52 * (YEARUN - 1))
      IF (CHGWK (1, I) .LT. ITEM) GO TO 7
C
C CHANGEWK HAS PASSED - USE 'B' CONVEN & CAPACITY
      $CONV = CONV (2,I)
      $CAP = CAP (2,I) - BCAP (2,I)
C
C
    7 IF ($CONV) 81112, 81112, 10
81112 TBKLOG (Q,I) = 0
      IF (Q .EQ. 4) TBKLOG (5,I) = 0
      GO TO 12
   10 IF ($CAP) 12, 12, 11
   11 TEM = $CAP
      BKL (Q) = (50. /$CONV) * (TBKLOG (Q, I) / TEM)
      IF (BKL (Q) .GT. 30) BKL (Q) = 30
      SBKLOG (Q) = SBKLOG (Q) + TBKLOG (Q,I)
      CBKLOG (Q) = CBKLOG (Q) + TBKLOG (Q,I)
      SBKL (Q) = SBKL (Q) + (BKL (Q) * TBKLOG (Q,I))
      CBKL (Q) = CBKL (Q) + (BKL (Q) * TBKLOG (Q,I))
   12 CONTINUE
      IF ($R1 .EQ. 0) GO TO 44444
C
      DO 20 Q = 1,5
      TEM = AOBCON
      AOB (Q) = TSDAYS (Q,I) / TEM
C MULT BY 4 IF QUARTERLY TOTAL
      IF (Q .LE. 4) AOB (Q) = AOB (Q) * 4.
      PASS (Q) = TUTIL (Q,I) - IFAIL (Q) - INADIS (Q)
      IF (PASS (Q) .LT. 0) PASS (Q) = 0
      PCTUTL (Q) = 0.
      TEM = TCAP (Q, I)
      IF (TCAP (Q, I) .GT. 0)
     1 PCTUTL (Q) = (TUTIL (Q, I) / TEM) * 100.
C
C TEST UTILIZATION TO SEE IF WE CAN CALCULATE PERCENTAGES
      IF (TUTIL (Q, I)) 13, 13, 14
C
C NO UTILIZATION - ZERO PERCENTAGES & EXIT
   13 CONTINUE
      PCTNSH (Q) = 0.
      PCTPAS (Q) = 0.
      PCTFL (Q) = 0.
      PCTSH (Q) = 0.
      PCTNDS (Q) = 0.
```

TABLE VI-17 (Continued)

KEND FORTRAN

```
      GO TO 20
C
C HAVE UTILIZATION - CALC *
   14 TEM = TUTIL (Q, I)
      TEM1 = TNOSHO (Q, I) + TUTIL (Q, I)
      PCTNSH (Q) = (TNOSHO (Q, I) / TEM1) * 100.
      PCTPAS (Q) = (PASS (Q) / TEM) * 100.
      PCTFL (Q) = (TFAIL (Q) / TEM) * 100.
      PCTSR (Q) = (TSETHK (Q, I) / TEM) * 100.
      PCTNDS (Q) = (INADIS (Q) / TEM) * 100.
   20 CONTINUE
C
C CONVERT LENGTH TO DAYS FOR PRINTOUT
      DO 27 Q = 1, 3
      IF (LEN (Q,I)) 28, 28, 26
   28  PLEN (Q) =  0. : GO TO 27
   26 ITEM = LEN (Q,I) - .05
      PLEN (Q) =(LEN (Q,I) * 5.) + (ITEM * 2) + .01
   27 CONTINUE
C
C CALL OUTPUT ROUTINES, CRT IF FLAG 0 OR 1, PRINT IF 1 OR 2
C SET X TO 1 FOR COURSE OUTPUT
C
      X = 1
      IF ($R1 .EQ. 0) GO TO 12341
      IF (YRFLG .LE. 1) CALL CRTOUT (X)
      IF (YRFLG .GE. 1) CALL NPRINT (X)
12341 CONTINUE
C
C IF I IS EQ CRSES THIS IS LAST COURSE, SO FALL THROUGH & PRINT SCHOOL
C SUMMARY.  OTHERWISE GO TO END OF DO LOOP & CONTINUE.
44444 IF (I .NE. CRSES) GO TO 4444
C
C ENTRY SCHOOL TOTALS CALC ROUTINE, SET X TO 2 FOR SCHOOL OUTPUT
C CALCULATE SCHOOL TOTALS (SAME ROUTINE USED FOR CENTER)
C
  500 X = 2
C
C 560 IS ENTRY FOR CENTER TOTALS CALCULATIONS
  560 CONTINUE
      IF (X .EQ. 2 .AND. $R2 .EQ. 0) GO TO 12342
      IF (X .EQ. 3 .AND. $R3 .EQ. 0) GO TO 12342
C
C CALCULATE YEARLY TOTALS
      DO 540 Q = 1, 4
      SCONV (5) = SCONV (5) + SCONV (Q)
      SCAP (5) = SCAP (5) + SCAP (Q)
      SUTIL (5) = SUTIL (5) + SUTIL (Q)
      SNOSHO (5) = SNOSHO (5) + SNOSHO (Q)
      SSDAYS (5) = SSDAYS (5) + SSDAYS (Q)
      SFAIL (5) = SFAIL (5) + SFAIL (Q)
      SSETHK (5) = SSETHK (5) + SSETHK (Q)
  540 SNADIS (5) = SNADIS (5) + SNADIS (Q)
C
C
```

TABLE VI-17 (Continued)

```
C NOW CALC PERCENTAGES
      DO 550 Q = 1, 5
C
      TEM = AOBCON
      AOM (Q) = SSDAYS (Q) / TEM
      IF (Q .LE. 4) AOM (Q) = AOB (Q) * 4.
C
      PASS (Q) = SUTIL (Q) - SFAIL (Q) - SNADIS (Q)
      IF (PASS (Q) .LT. 0) PASS (Q) = 0
C
      PCTUTL (Q) = 0.
      TEM = SCAP (Q)
      IF (TEM .GT. 0.) PCTUTL (Q) =(SUTIL (Q) / TEM) * 100.
C
      IF (SUTIL (Q)) 513, 513, 514
  513 CONTINUE
      PCTNSH (Q) = 0.
      PCTPAS (Q) = 0.
      PCTFL  (Q) = 0.
      PCTSH  (Q) = 0.
      PCTNDS (Q) = 0.
      GO TO 520
C
  514 TEM = SUTIL (Q)
      TEM1 = SNOSHO (Q) + SUTIL (Q)
      PCTNSH (Q) = (SNOSHO (Q) / TEM1) * 100.
      PCTPAS (Q) =(PASS (Q) / TEM) * 100.
      PCTFL (Q) = (SFAIL (Q) / TEM) * 100.
      PCTSH (Q) = (SSETBK (Q) / TEM) * 100.
      PCTNDS (Q) = (SNADIS (Q) / TEM) * 100.
  520 CONTINUE
      SHKL (Q) = SBKL (Q) / SHKLOG (Q)
  550 CONTINUE
C
C CALL OUTPUT SUBROUTINE, X IS 2 FOR SCHOOL OUT, 3 FOR CENTER OUT
C
      IF (YRFLG .LE. 1) CALL CRTOUT (X)
      IF (YRFLG .GE. 1) CALL NPRINT (X)
12342 CONTINUE
C
C TEST IF END OF FILE - CALC YEAR CENTER TOTALS
C TEST IF END OF FILE - GO PRINT CENTER TOTALS
      IF (ITEMS .EQ. CASES) GO TO 580
      IF (ITEMS .GT. 900) GO TO 4444
C
C
C KEEP RUNNING - ZERO SCHOOL TOTALS & RETURN
      DO 563 Q = 1, 5
      SUTIL (Q) = 0
      SCAP  (Q) = 0
      SSETBK(Q) = 0
      SSDAYS(Q) = 0
      SNOSHO(Q) = 0
      SCONV (Q) = 0
      SFAIL (Q) = 0
```

TABLE VI-17 (Continued)

AD-A040 238 IBM CORP CAPE CANAVERAL FLA F/G 5/9
DESIGN OF TRAINING SYSTEMS, TRAPAC USER'S MANUAL, DATA BASE, SC--ETC(U)
SEP 76 K V BRANCH, J D STALEY, R E YANKO N61339-73-C-0097
UNCLASSIFIED TAEG-36 NL

6 OF 6
AD A040238



END
DATE FILMED
7-77

KEND FORTRAN

```
      SBKL (Q) = 0
      SBKLOG (Q) = 0
  563 SNADIS(Q) = 0
C
      GO TO 4
C
C LOAD CENTER TOTALS INTO SCHOOL STORAGE  USE SCHOOL ROUTINE TO CALC
C CENTER TOTALS.
C
  580 DO 581 Q = 1, 5
      SUTIL (Q) = CUTIL (Q)
      SCAP (Q) = CCAP (Q)
      SSETBK (Q) = CSETBK (Q)
      SSDAYS (Q) = CSDAYS (Q)
      SNOSHO (Q) = CNOSHO (Q)
      SCONV (Q) = CCONV (Q)
      SFAIL (Q) = CFAIL (Q)
      SBKL (Q) = CBKL (Q)
      SBKLOG (Q) = CBKLOG (Q)
  581 SNADIS (Q) = CNADIS (Q)
C
C SET X TO 3 TO CALL CENTER OUTPUT, SET ITEM5 TO 999 TO INDICATE END
      ITEM5 = 999 ; X = 3 ; GO TO 560
C
 4444 CONTINUE
29999 CONTINUE
C EXIT IF YEARUN EQ Y AS LAST YEAR OF RUN
      IF (YEARUN .GE. Y) GO TO 6999
      YEAR = YEAR + 1
C
C SET TREND DEMANDS
      IF (TDMAND .LE. 0) GO TO 5555
      DO 5554 ITEM = 1, CRSES
      DO 5554 ITEM1 = 1, 3
      DEMAND(ITEM1, ITEM) = DEMAND (ITEM1, ITEM) * TDMAND
 5554 BDMAND (ITEM1, ITEM) = BDMAND (ITEM1, ITEM) * TDMAND
 5555 CONTINUE
C
C ZERO TOTALS FOR MULTI YEAR RUN
      DO 6998 I = 1, CRSES
      DO 6998 Q = 1, 4
      TUTIL (Q,I) = 0
      TCAP (Q,I) = 0
      TSETBK (Q,I) = 0
      TSDAYS (Q,I) = 0
      TNOSHO (Q,I) = 0
      TCONV (Q,I) = 0
      TFAIL (Q,I) = 0
 6998 TNADIS (Q,I) = 0
 6999 CONTINUE
      RETURN
      END
      SUBROUTINE NPRINT (X)
      IMPLICIT INTEGER*2 (A-Z,$)
C
```

TABLE VI-17 (Continued)

KEND FORTRAN

```
      RETURN
      END
      SUBROUTINE CRTOUT (X)
      IMPLICIT INTEGER*2 (A-Z,$)
      INTEGER*2 ILEN (3)
C COMMON FOR CLASS DATA MATRIX
      REAL*4 LEN, NOSHOW, NOSHPL, BKLGPL, BPOOL, NADIS, FAIL, CDP
      REAL*4 TDMAND, SETBAK, DATE
      COMMON /DATA/ CDP (328), CRSRUN (328), CONV (3,328), LEN (3,328)
      COMMON /DATA/ CAP (3,328), BCAP (3,328), CHGWK (2,328)
      COMMON /DATA/ OFFSET (328), DEMAND (3,328), BDMAND (3,328)
      COMMON /DATA/ BPOOL (328), FAIL (328), NADIS (328), NOSHPL (328)
      COMMON /DATA/ BAKLOG (328), NOSHOW (328), BKLGPL (328)
      COMMON /DATA/ BKLG (328), TDMAND, SETBAK, DATE
C
      INTEGER*4 PNEC
      REAL*4 PAOB, PFAIL
      REAL*8 NAME1, NAME2, CATNO1
C
      COMMON /TEXT/ PFAIL(328), NAME1 (328), NAME2 (328), CATNO1 (328)
      COMMON /TEXT/  CATNO2 (328), PNEC (328), TYPE (328), PAOB (328)
      COMMON /TEXT/ PRTY(328),SCH1(328),SCH2(328),SCH3(328),SCH4(328)
C
      REAL*4 TFAIL, TNADIS, DAYS (5)
      INTEGER*4 TSDAYS, TCAP, TUTIL
C
      COMMON /TOTALS/ TCAP (5, 328),   TUTIL (5, 328),  TSETBK (5, 328)
      COMMON /TOTALS/ TSDAYS (5, 328), TFAIL (5, 328),  TNOSHO (5, 328)
      COMMON /TOTALS/ TNADIS (5, 328), TBKLOG (5, 328), TCONV (5, 328)
C
C COMMON FOR SCHOOL AND CENTER TOTALS
C
      INTEGER*4 SUTIL, CUTIL, SCAP, CCAP, SSDAYS, CSDAYS, IFAIL, INADIS
      INTEGER*4 PASS, SBKLOG, CBKLOG, SBKL,CBKL
      REAL*4 AOB, PCTUTL, PCTNSH, PCTPAS, PCTFL, PCTSB, PCTNDS, PLEN
C
      COMMON /T/ SUTIL (5), SCAP (5), SSDAYS (5), SSETBK (5), SNOSHO (5)
      COMMON /T/ CUTIL (5), CCAP (5), CSDAYS (5), CSETBK (5), CNOSHO (5)
      COMMON /T/SBKL(5),SBKLOG(5),SCONV (5), SFAIL (5), SNADIS (5)
      COMMON /T/ CCONV(5),CBKL(5),CBKLOG(5), CFAIL (5), CNADIS (5)
      COMMON /T/ PCTUTL (5), PCTNSH (5), PCTNDS (5), INADIS (5)
      COMMON /T/ AOB (5), PASS (5), PCTPAS (5), PCTFL (5), PCTSB (5)
      COMMON /T/ IFAIL (5), PLEN (3), BKL (5), SPLAT (4), YEAR2D
C
C COMMON FOR COMMUNICATION BETWEEN SUBROUTINES
C
      INTEGER*4 ITEM, ITEM1, ITEM2, ITEM3, ITEM4, ITEM5, WHERE
      REAL*4 TEM, TEM1, TEM2, TEM3, TEM4, TEM5, $LEN
C
      COMMON DUMMY, $CONV, $LEN, $CAP, $BCAP, LAST, COURSE
      COMMON $R1,$R2,$R3,$R4, TEM, TEM1, TEM2, TEM3, TEM4, TEM5
      COMMON ITEM, ITEM1, ITEM2, ITEM3, ITEM4, ITEM5
      COMMON YEAR, CRSES, Q, W, WKOFF, INIT, WHERE, NEGFLG, AOBCON
      COMMON TRACEF, RLSFLG, ENDFLG, PREFLG, YRFLG,YEARUN,PRINTF,NGR
C
```

TABLE VI-17 (Continued)

KEND FORTRAN

```
C X IS 1 FOR COURSE OUTPUT, 2 FOR SCHOOL, 3 FOR CENTER (LAST)
C ITEMS CONTAINS COURSE POINTER
C
      ILEN(1) = PLEN(1); ILEN(2)=PLEN(2); ILEN(3)=PLEN(3)
       IF (PRINTF .GT.X) RETURN
      DFL = 0 ;/* INIT DAYS GT 99999 FLAG */
      DO 10 I = 1,4
      DAYS (I) = SSDAYS (I) / 1000.
   10 IF (SSDAYS(I) .GE. 99999) DFL = 1
      DAYS (5) = SSDAYS (5) / 1000.
       I = ITEMS ; IF (X .EQ. 1) GO TO 1
C
      WRITE (11, 5001) YEAR
 5001 FORMAT ('1', T22, 'TPF MODEL --- FISCAL YEAR ', I4, ' SUMMARY')
C
      IF (X .EQ. 2) GO TO 5000
C
C CRT ROUTINE FOR CENTER TOTALS - ACTUAL DATA OUTPUT IN SCHOOL ROUTINE
      WRITE (11, 6001)
 6001 FORMAT (' ', T21, '** TOTALS FOR  TRAPAC TRAINING CENTER **' /)
      IF ($R2 .LE. 1) GO TO 16002
      WRITE (11,6002) SCH1(CRSES),DATE;/* SEL OPTION- PRINT 1ST CHAR */
 6002 FORMAT (' CENTER CODE  ',A1,'  SUMMARY (RUN ',A4,'76)'/)
      GO TO 15000
16002 WRITE (11,26002) DATE  ;/* NO SEL OPTION- TRAPAC TOTALS */
26002 FORMAT (' TRAPAC TOTALS (RUN ',A4,'76)'/)
      GO TO 15000
C
C OUTPUT FORMATS FOR CRT DISPLAY, COMPATABLE WITH OUTPUT FORMAT PROGRAM
C FORMAT - ALWAYS 25 LINES OF OUTPUT, 1 FOR FIRST LINE, BLANK FOR DATA
C LINES, / FOR LAST LINE, * FOR END OF DATA
C
    1 WRITE (9, 4001) CDP (I), NAME1 (I), NAME2 (I)
      WRITE (9, 4002) CONV (1,I), CONV (2,I), CONV (3,I), BKLG(I)
      WRITE (9, 4003) ILEN (1), ILEN (2), ILEN (3), BAKLOG, (I)
      WRITE (9, 4004) SCH1(I),SCH2(I),SCH3(I),SCH4(I), DATE,
     * CATNO1(I),CATNO2(I),PNEC (I), CAP (1,I), CAP (2,I), CAP (3,I)
      WRITE (9, 4005) BCAP (1,I), BCAP (2,I), BCAP (3,I), CHGWK (1,I)
      WRITE (9, 4006) DEMAND (1,I), DEMAND (2,I), DEMAND (3,I),
     * CHGWK (2,I)
      WRITE (9, 4007) PAOB (I), PFAIL (I), TYPE (I), PRTY (I),
     * BDMAND (1,I), BDMAND (2,I), BDMAND (3,I), OFFSET (I)
      WRITE (9, 4008) SPLAT (1), SPLAT (2), SPLAT (3)
      WRITE (9, 4009) YEAR2D, YEAR2D, YEAR2D, YEAR2D
      WRITE (9, 4010)
      WRITE (9, 4011) TCONV (1,I), TCONV (2,I), TCONV (3,I), TCONV (4,I)
     1 , TCONV (5,I)
      WRITE (9, 4012) TCAP (1,I), TCAP (2,I), TCAP (3,I), TCAP (4,I),
     1 TCAP (5,I)
      WRITE (9, 4013) ((TUTIL (Q,I), PCTUTL (Q)), Q = 1, 5)
      WRITE (9, 4014) ((TNOSHO (Q,I), PCTNSH (Q)), Q = 1, 5)
      WRITE (9, 4015) ((TBKLOG (Q,I), BKL (Q)), Q = 1, 5)
      WRITE (9, 4016) ((TSDAYS (Q, I), AOB (Q)), Q = 1, 5)
      WRITE (9, 4017)
      WRITE (9, 4018) ((PASS (Q), PCTPAS (Q)), Q = 1, 5)
```

KEND FORTRAN

```
      WRITE (9, 4019) ((IFAIL (Q), PCTFL (Q)), Q = 1, 5)
      WRITE (9, 4020) ((ISETBK (Q,I), PCTSB (Q)), Q = 1, 5)
      WRITE (9, 4021) (( INADIS (Q), PCTNDS (Q)), Q = 1, 5)
      WRITE (9, 4022)
      RETURN
C CRT ROUTINE FOR SCHOOL AND CENTER TOTALS, ENTRY IS 15000 FOR CENTER
 5000 IF ($R2 .EQ. 2) GO TO 25000
      WRITE (11,5002) SCH1(I)  ;/* NO SEL OPTION- JUST 1ST CHAR SCH */
 5002 FORMAT (1X,T34,'SCHOOL CODE  ',A1/)
      WRITE (11,5003)  SCH1(I), DATE
 5003 FORMAT (1X,A1,2X,'SCHOOL SUMMARY  (RUN ',A4,'76)'/)
      GO TO 15000
25000 WRITE (11,15002) SCH1(I),SCH2(I),SCH3(I),SCH4(I)
15002 FORMAT (1X,T33,'SCHOOL CODE  ',4A1/)
      WRITE (11,15003) SCH1(I),SCH2(I),SCH3(I),SCH4(I),DATE
15003 FORMAT (1X,4A1,2X,'SCHOOL SUMMARY (RUN ',A4,'76)'/)
C
C ENTRY CENTER TOTALS
C
15000 WRITE (11, 4009) YEAR2D, YEAR2D, YEAR2D, YEAR2D
      WRITE (11, 4010)
      WRITE (11, 5004) SCONV(1), SCONV(2), SCONV(3), SCONV(4), SCONV(5)
      WRITE (11, 5005) SCAP (1), SCAP (2), SCAP (3), SCAP (4), SCAP (5)
      WRITE (11, 5006) ((SUTIL (Q), PCTUTL (Q)), Q = 1, 5)
      WRITE (11, 5007) ((SNOSHO (Q), PCTNSH (Q)), Q = 1, 5)
      WRITE (11,14015) (SPKLOG (Q),SBKL(Q),Q=1,5)
      IF(DFL.EQ.0)WRITE (11, 5008) ((SSDAYS (Q), AOB (Q)), Q = 1, 5)
      IF(DFL.EQ.1)WRITE (11,15008) ((DAYS(Q),AOB(Q)),Q=1,5)
      WRITE (11, 4017)
      WRITE (11, 5009) ((PASS (Q), PCTPAS (Q)), Q = 1, 5)
      WRITE (11, 5010) ((SFAIL (Q), PCTFL (Q)), Q = 1, 5)
      WRITE (11, 5011) ((SSETBK (Q), PCTSB (Q)), Q = 1, 5)
      WRITE (11, 5012) ((SNADIS (Q), PCTNDS (Q)), Q = 1, 5)
      WRITE (11, 4022)
      RETURN
C
C FORMATS FOR 80 COLUMN CRT OUTPUT
C
 4001 FORMAT ('1', A4, 5X, 2A4, 4X, '--SCHEDULE DATA--   FY   FY+1   FY+
     *2   CRS BACKLOG')
C
 4002 FORMAT (' ', T31, 'ANNUAL CONVENINGS', I5, 2I7, 4X, '# STU', I5)
C
 4003 FORMAT (' SCHL DATE CATALOG NO  NEC    COURSE LENGTH DYS', I5,
     * 2I7, 4X, '# WKS', I5)
C
 4004 FORMAT (' ', 4A1,1X,A4,1X,A4, A2, 1X, A4, 4X, 'COURSE CAPACITY',
     * 3I7)
C
 4005 FORMAT (' ', T31, 'RUPERS CAPACITY', 3I7, 3X, 'FY+1 WK', I4)
C
 4006 FORMAT (' P AOB  PFAIL  TYPE   PRTY    ANNUAL DEMAND', 2X, 3I7,
     * 3X, 'FY+2 WK', I4)
C
 4007 FORMAT (' ', F5.1, F6.1, '%', 3X, 2 (A2, 5X), 'RUPERS DEMAND',
```

TABLE VI-17 (Continued)

KEND FORTRAN

```
     * 2X, 317, 3X, 'OFFSET', I5)
C
 4008 FORMAT (1X, T46, 3 (5X, A2), / 1X, T33, 'QUARTERLY PROJECTIONS')
C
 4009 FORMAT (' ', 12X, '| 1 QR FY', I3, ' | 2 QR FY', I3,
     1 ' | 3 QR FY', I3, ' | 4 QR FY', I3, ' | ANNUAL TOTAL |')
C
 4010 FORMAT (' ', '------------|------------|------------|',
     1 '------------|------------|-------------|')
C
 4011 FORMAT (' ', '# CONVENINGS|', I4, ' |', 5X,    ' |', I4, ' |',
     1 5X,    ' |', I4, ' |', 5X,  ' |', I4, ' |', 5X,    ' |',
     2 I6, ' |', 5X,    ' |')
C
 4012 FORMAT (' ', 'CRS CAPACITY|', I4, ' |', 5X,    ' |', I4, ' |',
     1 5X,    ' |', I4, ' |', 5X,  ' |', I4, ' |', 5X,    ' |',
     2 I6, ' |', 5X,    ' |')
C
 4013 FORMAT (' ', 'UTILIZATION |', I4, ' |', F5.1, '%|', I4, ' |',
     1 F5.1, '%|', I4, ' |', F5.1,'%|', I4, ' |', F5.1, '%|',
     2 I6, ' |', F5.1, '%|')
C
 4014 FORMAT (' ', 'NO SHOWS    |', I4, ' |', F5.1, '%|', I4, ' |',
     1 F5.1, '%|', I4, ' |', F5.1,'%|', I4, ' |', F5.1, '%|',
     2 I6, ' |', F5.1, '%|')
C
 4015 FORMAT (' ', 'BAKLOG #/WKS|', I4, ' |', I2,' WKS|', I4, ' |',
     1 I2,' WKS|', I4, ' |', I2,' WKS|',I4, ' |', I2,' WKS|',
     2 I6, ' |', I2,' WKS|')
C
 4016 FORMAT (' ', 'STU DAYS/AOB|', I4, ' |', F5.1, ' |', I4, ' |',
     1 F5.1, ' |', I4, ' |', F5.1,' |', I4, ' |', F5.1, ' |',
     2 I6, ' |', F5.1, ' |')
C
 4017 FORMAT (' ', 12X, '|', 5X, '|', 6X, '|', 5X, '|', 6X, '|', 5X,
     1 '|', 6X, '|', 5X, '|', 6X, '|', 7X, '|', 6X, '|')
C
 4018 FORMAT (' ', 'TOTAL PASS  |', I4, ' |', F5.1, '%|', I4, ' |',
     1 F5.1, '%|', I4, ' |', F5.1,'%|', I4, ' |', F5.1, '%|',
     2 I6, ' |', F5.1, '%|')
C
 4019 FORMAT (' ', 'TOTAL FAIL  |', I4, ' |', F5.1, '%|', I4, ' |',
     1 F5.1, '%|', I4, ' |', F5.1,'%|', I4, ' |', F5.1, '%|',
     2 I6, ' |', F5.1, '%|')
C
 4020 FORMAT (' ', 'SETBACK ACAD|', I4, ' |', F5.1, '%|', I4, ' |',
     1 F5.1, '%|', I4, ' |', F5.1,'%|', I4, ' |', F5.1, '%|',
     2 I6, ' |', F5.1, '%|')
C
 4021 FORMAT (' ', 'DISENROLL NA|', I4, ' |', F5.1, '%|', I4, ' |',
     1 F5.1, '%|', I4, ' |', F5.1,'%|', I4, ' |', F5.1, '%|',
     2 I6, ' |', F5.1, '%|')
C
 4022 FORMAT (' ', '-----------------------------------------',
     *         '-----------------------------------------',
```

KEND FORTRAN

```
     2 /'/'/' ')
C
C SCHOOL AND CENTER CRT FORMATS
C
 5004 FORMAT (' ', '# CONVENINGS!', I5, '!', 5X, ' !', I5, '!',
     1 5X, ' !', I5, '!', 5X, ' !', I5, '!', 5X, ' !',
     2 I6, ' !', 5X, ' !')
C
 5005 FORMAT (' ', 'CRS CAPACITY!', I5, '!', 5X, ' !', I5, '!',
     1 5X, ' !', I5, '!', 5X, ' !', I5, '!', 5X, ' !',
     2 I6, ' !', 5X, ' !')
C
 5006 FORMAT (' ', 'UTILIZATION !', I5, '!', F5.1, '%!', I5, '!',
     1 F5.1, '%!', I5, '!', F5.1,'%!', I5, '!', F5.1, '%!',
     2 I6, ' !', F5.1, '%!')
C
 5007 FORMAT (' ', 'NO SHOWS !', I5, '!', F5.1, '%!', I5, '!',
     1 F5.1, '%!', I5, '!', F5.1,'%!', I5, '!', F5.1, '%!',
     2 I6, ' !', F5.1, '%!')
C
14015 FORMAT (' ', 'MAKLOG #/WKS!', I5, '!', I2,' WKS!', I5, '!',
     1 I2,' WKS!', I5, '!', I2,' WKS!',I5, '!', I2,' WKS!',
     2 I6, ' !', I2,' WKS!')
C
 5008 FORMAT (' ', 'STU DAYS/AOB!', I5, '!', F6.1, '!', I5, '!',
     1 F6.1, '!', I5, '!', F6.1, '!', I5, '!', F6.1, '!',
     2 I6, ' !', F6.1, '!')
C
15008 FORMAT(' KSTU DAY/AOB!',4(F5.1,'!',F6.1,'!'),F6.1,' !',F6.1,'!')
C
 5009 FORMAT (' ', 'TOTAL PASS !', I5, '!', F5.1, '%!', I5, '!',
     1 F5.1, '%!', I5, '!', F5.1,'%!', I5, '!', F5.1, '%!',
     2 I6, ' !', F5.1, '%!')
C
 5010 FORMAT (' ', 'TOTAL FAIL !', I5, '!', F5.1, '%!', I5, '!',
     1 F5.1, '%!', I5, '!', F5.1,'%!', I5, '!', F5.1, '%!',
     2 I6, ' !', F5.1, '%!')
C
 5011 FORMAT (' ', 'SETBACK ACAD!', I5, '!', F5.1, '%!', I5, '!',
     1 F5.1, '%!', I5, '!', F5.1,'%!', I5, '!', F5.1, '%!',
     2 I6, ' !', F5.1, '%!')
C
 5012 FORMAT (' ', 'DISENROLL NA!', I5, '!', F5.1, '%!', I5, '!',
     1 F5.1, '%!', I5, '!', F5.1,'%!', I5, '!', F5.1, '%!',
     2 I6, ' !', F5.1, '%!')
C
      RETURN
      END
      SUBROUTINE RANGE (KK,LL)
      IMPLICIT INTEGER*2 (A-Z,$)
      INTEGER*4 ITEM, ITEM1, ITEM2, ITEM3, ITEM4, ITEM5, WHERE
      REAL*4 TEM, TEM1, TEM2, TEM3, TEM4, TEM5, $LEN
      COMMON DUMMY, $CONV, $LEN, $CAP, $BCAP, LAST, COURSE
      COMMON $R1,$R2,$R3,$R4, TEM, TEM1, TEM2, TEM3, TEM4, TEM5
      COMMON ITEM, ITEM1, ITEM2, ITEM3, ITEM4, ITEM5
```

KEND FORTRAN

```
      COMMON YEAR, CRSES, Q, W, WKOFF, INIT, WHERE, NEGFLG, AOBCON
      COMMON TRACEF, RLSFLG, ENDFLG, PREFLG, YRFLG,YEARUN,PRINTF,NGR
C RANGE ERROR SUBROUTINE FOR DIRECT ACCESS IO
      NGR = 10 ; KK = 0 ; LL = 232
      RETURN
      END
      BLOCK DATA
C INITIAL VALUES FOR PARAMETERS IN COMMON STORAGE
      IMPLICIT INTEGER*2 (A-Z,$)
      REAL*4 LEN, NOSHOW, NOSHPL, BKLGPL, BPOOL, NADIS, FAIL, CDP
      REAL*4 TDMAND, SETBAK, DATE
C
      COMMON /DATA/ CDP (328), CRSRUN (328), CONV (3,328), LEN (3,328)
      COMMON /DATA/ CAP (3,328), RCAP (3,328), CHGWK (2,328)
      COMMON /DATA/ OFFSET (328), DEMAND (3,328), BDMAND (3,328)
      COMMON /DATA/ BPOOL (328), FAIL (328), NADIS (328), NOSHPL (328)
      COMMON /DATA/ BAKLOG (328), NOSHOW (328), BKLGPL (328)
      COMMON /DATA/ BKLG (328), TDMAND, SETBAK, DATE
C
      DATA BKLG/326*0/, CRSRUN /328*0/, BKLGPL, NOSHPL, BPOOL /984*0./
C
C QUARTERLY AND ANNUAL COURSE TOTALS ACCUMULATION
      REAL*4 TFAIL, TNADIS
      INTEGER*4 TSDAYS, TCAP, TUTIL
C
      COMMON /TOTALS/ TCAP (5, 328),   TUTIL (5, 328),  TSETBK (5, 328)
      COMMON /TOTALS/ TSDAYS (5, 328), TFAIL (5, 328),  TNOSHO (5, 328)
      COMMON /TOTALS/ TNADIS (5, 328), TBKLOG (5, 328), TCONV (5, 328)
C
      DATA TCAP /1640*0/, TUTIL /1640*0/, TSETBK /1640*0/
      DATA TNADIS /1640*0./, TBKLOG /1640*0/, TCONV /1640*0/
      DATA TSDAYS /1640*0/, TFAIL /1640*0./, TNOSHO /1640*0/
C
C COMMON FOR COURSE ACTIVE MATRIX ELEMENTS
C
      REAL*4 ACTCTR
      COMMON /RUN/ ACTIVE (250), ACTCTR (250), ACTUTL (250), ACTNEG(250)
C
      DATA ACTIVE /250*0/, ACTCTR /250*0./, ACTUTL, ACTNEG /500*0/
C
C COMMON FOR SCHOOL AND CENTER TOTALS
C
      INTEGER*4 SUTIL, CUTIL, SCAP, CCAP, SSDAYS, CSDAYS, IFAIL, INADIS
      INTEGER*4 PASS, SBKLOG, CBKLOG, SBKL,CBKL
      REAL*4 AOB, PCTUTL, PCTNSH, PCTPAS, PCTFL, PCTSB, PCTNDS, PLEN
C
```

TABLE VI-17 (Continued)

KEND FORTRAN

```
      COMMON /T/ SUTIL (5), SCAP (5), SSDAYS (5), SSETBK (5), SNOSHO (5)
      COMMON /T/ CUTIL (5), CCAP (5), CSDAYS (5), CSETBK (5), CNOSHO (5)
      COMMON /T/SBKL(5),SBKLOG(5),SCONV (5), SFAIL (5), SNADIS (5)
      COMMON /T/ CCONV(5),CBKL(5),CBKLOG(5), CFAIL (5), CNADIS (5)
      COMMON /T/ PCTUTL (5), PCTNSH (5), PCTNDS (5), INADIS (5)
      COMMON /T/ AOH (5), PASS (5), PCTPAS (5), PCTFL (5), PCTSB (5)
      COMMON /T/ IFAIL (5), PLEN (3), BKL (5), SPLAT (4), YEAR2D
C
      DATA SPLAT /' *',3*'  '/
C
      END
```

TABLE VI-17 (Continued)

SECTION VII

SYSTEM CAPABILITIES/REQUIREMENTS AND RESOURCES MODEL

PROGRAM DESCRIPTIONS

MODEL DESCRIPTION

The SCRR model is a set of nineteen (19) VP/CSS Executive procedures, twelve (12) FORTRAN programs, three (3) PL/1 programs, and one (1) MPS III/OL procedure. All of these procedures and programs are executed under the control of the nine (9) primary VP/CSS Executives depicted in Figure VII-1. Figure VII-1 shows the overall flow through the nine executives and the connection with the RAMIS data base. The various input and output data files and all executives, programs and procedures executed by the primary executives do not appear, since some are accessed several times and the overview flowchart would be nearly unintelligible.

Table VII-1 gives a brief description of the VP/CSS Executive programs utilized by the SCRR model. All RAMIS procedures, FORTRAN programs, PL/1 programs, and the MPS III/OL procedure are executed by these executives. This section of the User's Manual has been organized based on these VP/CSS executive programs. For example, the RAMSCRR subsection contains the verbal descriptions and listings for the RAMSCRR EXEC and the six RAMIS procedures executed by RAMSCRR. The SCRR1 subsection contains descriptions and listings for the SCRR1 EXEC, the MPS III/OL procedure, and the FORTRAN subroutine executed under control of MPS III.

Table VII-2 gives a list of the input and output data files used by each of the executive programs. All files are stored on disk and are named by a unique three-part identifier composed of a filename, a filetype and a filemode. Only the filename and filetype are shown in Table VII-2. All these files have a filemode of P which indicates that they reside on the P (Permanent) disk.

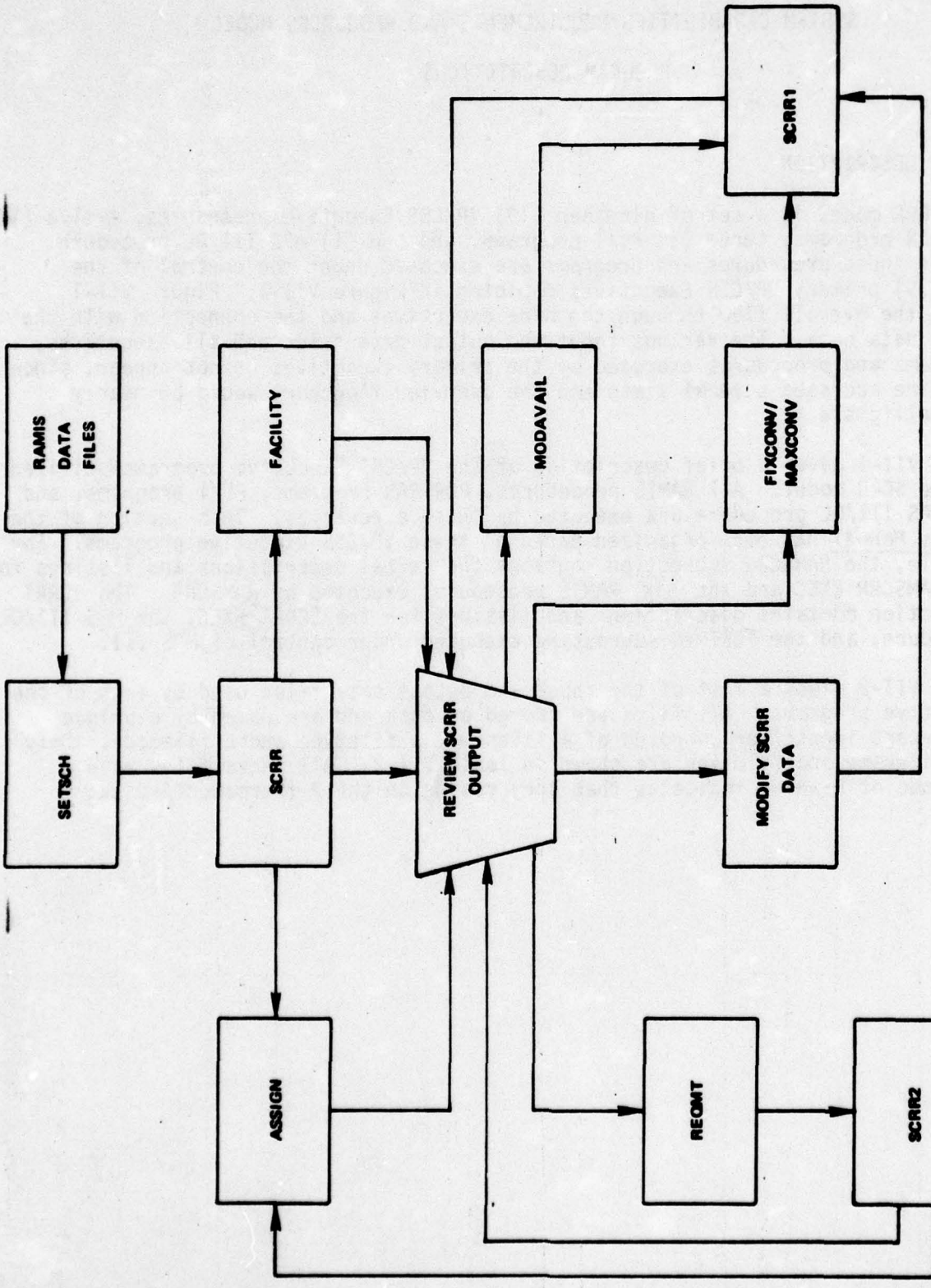

OVERVIEW OF SCRR EXECUTIVE PROGRAMS
RAMIS DATA FILES
FACILITY
MODAVAIL
FIXCONV/ MAXCONV
SCRR1
SETSCH
SCRR
REVIEW SCRR OUTPUT
MODIFY SCRR DATA
ASSIGN
REOMT
SCRR2

FIGURE VII-1

SCRR EXECUTIVE PROGRAMS DESCRIPTION

EXEC NAME	DESCRIPTION
SETSCH	Initializes the RAMIS data base procedures to retrieve data for only the specified organization.
SCRR	A top level executive which executes the following programs: RAMSCRR,BLANK,IFI,IF2,SCRRSORT, IF3,IF4,COMB,and SCRRI.
RAMSCRR	Uses the RAMIS data base procedures SCRR INSTI, SCRR INST2,SCRR FACIL,SCRR IAVAIL, SCRR FAVAIL,and SCRR CRSE to retireve all all SCRR model basic input data from the three data files COURSE,INST,and FACIL and to derive additional data from these basic inputs.
BLANK	Deletes all internal blank characters from facility names in the files FILE FACIL and FILE FAVAIL.
IF1	Formats NAME and ROW data for LP input.
IF2	Formats LP COLUMN data including instructor and facility requirements and objective function coefficients.
SCRRSORT	Sorts IF2 output by course CDP number.
IF3	Formats LP RHS data section, including instructor and facility availabilities.
IF4	Formats BOUNDS data section, including course convenings limits, of LP input data.
COMB	Combines outputs from IFI,IF2,IF3,and IF4 into a single file SCRR DATA which is directly readable through MPS III;OL Terminal Control Language
SCRRI	Establishes all control data (i.e., input, output, and internal processing files) required by MPS III/OL system and initiates the LP processing.
SCRR2	A top level executive which executes the following programs: BLANK, IF2, SCRRSORT, COMB, SCRR1.
SCRR3	A top level executive which executes the following programs: BLANK,IFI,IF2,SCRRSORT,IF3,IF4, COMB, and SCRRI.

TABLE IV-1

SCRR EXECUTIVE PROGRAMS DESCRIPTION

EXEC NAME	DESCRIPTION
ASSIGN	Compiles and prints a list of instructor assignments and contact hour requirements.
FACILITY	Compiles and prints a list of facility utilization data.
MODAVAIL	Modifies resource availability data prior to LP optimization.
REQMT	Permits user to specify percentage changes to variables which impact instructor requirements.
FIXCONV	Modifies the LP input data to fix the number of course convenings at the current level.
MAXCONV	Modifies the LP input data to optimize the number of course convenings.

TABLE IV-1 (Continued)

SCRR DATA FILE SUMMARY

EXEC	INPUT FILES	OUTPUT FILES
RAMSCRR	RAMIS files COURSE, INST, FACIL	FILE INST, FILE IAVAIL, FILE FACIL, FILE FAVAIL, FILE CRSE
BLANK	FILE FACIL, FILE FAVAIL	FILE FACIL, FILE FAVAIL
IF1	FILE IAVAIL, FILE FAVAIL	FILE IF1
IF2	FILE INST, FILE FACIL, FILE CRSE	FILE IF2
SCRRSORT	FILE IF2	FILE IF2
IF3	FILE IAVAIL, FILE FAVAIL	FILE IF3
IF4	FILE CRSE	FILE IF4
COMB	FILE IF1, FILE IF2, FILE IF3, FILE IF4	SCRR DATA
SCRR1	SCRR DATA, SCRR RUN, SCRRFORM LNKLIB	SCRR OUTPUT
ASSIGN	FILE INST, FILE CRSE, FILE IAVAIL	ASSIGN LISTING
FACILITY	FILE FACIL,FILE CRSE, FILE FAVAIL	FACILITY LISTING
MODAVAIL	SCRR DATA	SCRR DATA
REQMT	RAMIS files COURSE and FACIL	FILE FACIL, FILE INST, FILE CRSE
FIXCONV	SCRR DATA	SCRR DATA
MAXCONV	SCRR DATA	SCRR DATA

TABLE IV-2

SETSCH EXEC DESCRIPTION

The purpose of the executive SETSCH is to identify the organization for which data is to be retrieved from the RAMIS data files. SETSCH uses the CSS EDIT command to effect the required changes in the RAMIS procedures SCRR INST1, SCRR INST2, SCRR FACIL, SCRR CRSE, SCRR IAVAIL, and SCRR FAVAIL. The EDIT command copies the RAMIS procedure into core. Subsequent edit requests examine and modify the core image of the procedure and then write the procedure to disk.

The initial EXEC statement suppresses printing of all CSS command statements and all EXEC control statements. Next, the output file for the FORTRAN program SETSCH is specified and SETSCH is executed. The EXEC file SS which has been created by SETSCH is then executed. SS EXEC uses the CSS EDIT command to modify the RAMIS procedures. First, the line containing the phrase "if sch is" is located in the procedure by the locate request. Then the four-digit activity/department code specified by the user is overlayed in the positions following the phrase "if sch is" using the EDIT command OVERLAY request. The final statement in the SS EXEC writes the updated RAMIS procedures to a writable disk. SS EXEC is erased following execution.

SETSCH FORTRAN DESCRIPTION. The purpose of SETSCH FORTRAN is to create the SS EXEC file which contains the statements required to include the specified activity/department code in the RAMIS data base procedures.

The user specifies the four digit activity/department code. This code is formatted within several CSS EDIT command request statements and written on output file SS EXEC. The SS EXEC file created by SETSCH is shown below:

SS EXEC

```
&STACK KT
&BEGSTACK
L. /IF SCH IS/
O.                ASW1
FILE
&ENDSTACK
EDIT SCRR FACIL
&STACK KT
&BEGSTACK
L. /IF SCH IS/
O.                ASW3
FILE
&ENDSTACK
EDIT SCRR FAVAIL
&STACK KT
&BEGSTACK
L. /IF SCH IS/
O.                ASW4
FILE
&ENDSTACK
EDIT SCRR CRSE
```

SS EXEC (Continued)

```
&STACK KT
&BEGSTACK
L. /IF SCH IS/
0.                ASW4
FILE
&ENDSTACK
EDIT SCRR IAVAIL
&STACK KT
&BEGSTACK
L. /IF SCH IS/
0.                ASW5
FILE
&ENDSTACK
EDIT SCRR INST1
&STACK KT
&BEGSTACK
L. /IF SCH IS/
0.                ASW6
FILE
&ENDSTACK
EDIT SCRR INST2
```

SETSCH PROGRAM LISTINGS

SETSCH EXEC

```
&TYPE OFF
FILEDEF 6 DSK SS EXEC
FILEDEF 6 CONO
FILEDEF 5 CON
RUN SETSCH
EXEC SS
ERASE SS EXEC
```

SETSCH FORTRAN

```
      INTEGER*4 SCH
      WRITE (6,9)
    9 FORMAT('ENTER 4 DIGIT SCHOOL CODE:')
      READ(5,10) SCH
   10 FORMAT (A4)
      WRITE (8,11) SCH,SCH,SCH,SCH,SCH,SCH
   11 FORMAT ('&STACK KT'/'&BEGSTACK'/'L. /IF SCH IS'/'0.          ',
     1A4/'FILE'/'&ENDSTACK'/'EDIT SCRR FACIL'/
     1'&STACK KT'/'&BEGSTACK'/'L. /IF SCH IS'/'0.          ',
     1A4/'FILE'/'&ENDSTACK'/'EDIT SCRR FAVAIL'/
     1'&STACK KT'/'&BEGSTACK'/'L. /IF SCH IS'/'0.          ',
     1A4/'FILE'/'&ENDSTACK'/'EDIT SCRR CRSE'/
     1'&STACK KT'/'&BEGSTACK'/'L. /IF SCH IS'/'0.          ',
     1A4/'FILE'/'&ENDSTACK'/'EDIT SCRR IAVAIL'/
     1'&STACK KT'/'&BEGSTACK'/'L. /IF SCH IS'/'0.          ',
     1A4/'FILE'/'&ENDSTACK'/'EDIT SCRR INST1'/
     1'&STACK KT'/'&BEGSTACK'/'L. /IF SCH IS'/'0.          ',
     1A4/'FILE'/'&ENDSTACK'/'EDIT SCRR INST2')
      END
```

SCRR EXEC DESCRIPTION

The purpose of executive SCRR is to execute all SCRR model-related executives through a single keyboard entry.

Upon entry, a test is made to determine if core size is equal to or greater than 320K. If core size is less than 320K, a message directing the user to SET CORE 320 is issued and SCRR EXEC is terminated. If the core requirement is met, a message describing the objective of the EXEC is printed and the user is given the alternative of proceeding with model execution or terminaing the SCRR EXEC at this point. If the user elects to proceed with SCRR model execution, the following EXECs are executed serially:

RAMSCRR
BLANK
IF1
IF2
SCRRSORT
IF3
IF4
COMB
SCRR1

Each of these executives is described in detail on the following pages.

SCRR EXEC FLOWCHART

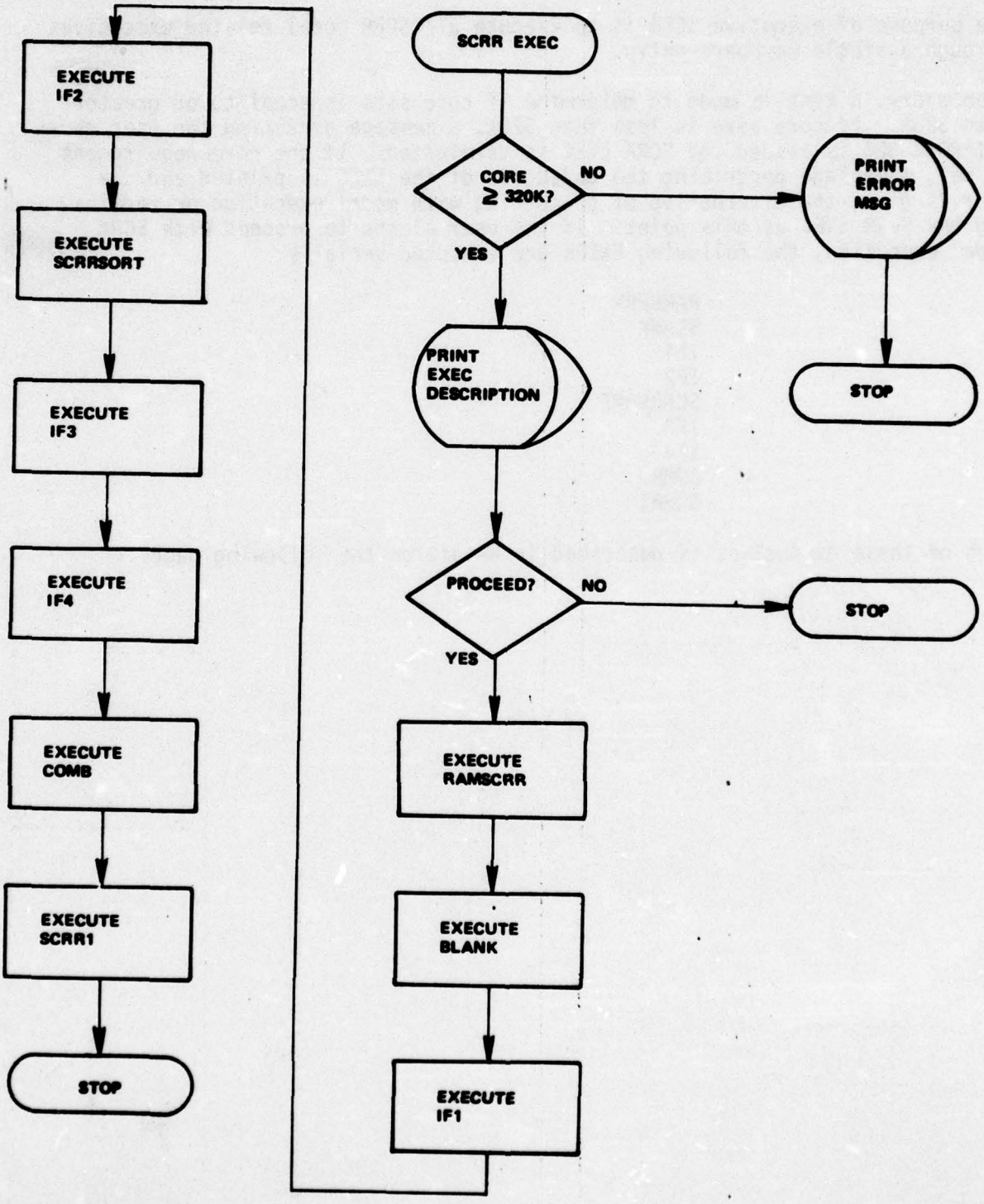


FIGURE VII-2

SCRR PROGRAM LISTING

SCRR EXEC

```
&IF &CORE GE 320 &SKIP 3
&PRI REQUIRED CORE IS 320K. PLEASE TYPE: SET CORE 320
&PRI THEN REISSUE COMMAND: SCRR
&EXIT
&PRI THIS PROGRAM PERFORMS THE LINEAR PROGRAMMING OPTIMIZATION
&PRI OF THE RESOURCE DATA CURRENTLY STORED IN THE MASTER DATA FILES
&PRI FOR THE DEPARTMENT IDENTIFIED THROUGH EXECUTION OF THE SETSCH
&PRI COMMAND. NO FURTHER USER INTERVENTION WILL BE REQUIRED.
&PRI DO YOU WISH TO PROCEED? (Y/N)
&READ ARGS
&IF &1 NE Y &GOTO -EXIT
&STACK KT
EXEC RAMSCRR
EXEC BLANK
EXEC IF1
EXEC IF2
EXEC SCRRSORT
&STACK KT
EXEC IF3
EXEC IF4
EXEC COMP
EXEC SCRR1
-EXIT &EXIT
```

TABLE VII-3

RAMSCRR EXEC DESCRIPTION

All interface between the SCRR model and the DOTS data base is performed by the RAMSCRR EXEC. RAMSCRR utilizes a series of RAMIS data base procedures to retrieve (and to calculate when required) instructor and facility requirements and availabilities and course parameters from the three RAMIS data files INST, COURSE, and FACIL.

Initial RAMSCRR statements suppress all terminal output and gain access to RAMIS data base software. Then the output file FILE INST1 is established for the RAMIS procedure SCRR INST1, and SCRR INST 1 is executed. Similarly, output files FILE INST2, FILE FACIL, FILE FAVAIL, FILE IAVAIL, and FILE CRSE are defined and RAMIS procedures SCRR INST2, SCRR FAIL, SCRR FAVAIL, SCRR IAVAIL, and SCRR CRSE are executed. The files FILE INST1 and FILE INST2 are combined by the FORTRAN program SCRRINST to form the file FILE INST. RAMSCRR EXEC concludes by removing the RAMIS software disk from the user's virtual machine.

SCRR INST1 AND SCRR INST2 DESCRIPTIONS. The purpose of the RAMIS procedures SCRR INST1 and SCRR INST2 is to calculate instructor requirements in hours by course and by instructor. The basic data required to determine instructor requirements are stored in the RAMIS course file COURSE.

SCRR INST1 first retrieves the required data fields for the activity/department code specified by the user in the SETSCH EXEC. The total number of contact hours required (TOTHRS) for each student/instructor ratio (F1) within each course (CDP) is calculated. The class capacity is multiplied by the historical course utilization percentage to determine average class size. Average class size is divided by the student/instructor ratio (F1) to determine the total number of instructors (NOINST) required at each student/instructor ratio. NOINST is multiplied by the required contact hours at each ratio to arrive at TOTHRS. TOTHRS are then summed by CDP to get the total contact hours required for each course. The number of instructors physically assigned to each course is determined by summing the value of F3 (1=assigned, 0=not assigned). TOTHRS is then divided by the sum of F3 to determine the total number of hours required per instructor per course. Output is written on FILE INST1. The record format for FILE INST1 is shown below:

FIELD NAME	CDP NO.	REQ'D HRS.
TYPE	A	F
LENGTH	4	5-1
RELATIVE POS.	1-4	5-9

SCRR INST2 determines the specific instructors by ID and name assigned to each course. Output is written on FILE INST2. The record format for FILE INST2 is shown below:

FIELD NAME	CDP NO.	INST ID	INST NAME
TYPE	A	I	A
LENGTH	4	9	16
RELATIVE POS.	1-4	5-13	14-29

The FORTRAN program SCRRINST combines records from the two files FILE INST1 and FILE INST2. Required contact hours per instructor per course from FILE INST1 is appended to the appropriate record in FILE INST2 containing course numbers and instructor identification. The combined records are written on FILE INST. The record format for FILE INST is shown below:

FIELD NAME	CDP NO.	INST ID	INST NAME	REQ'D HRS.
TYPE	A	I	A	F
LENGTH	4	9	16	5.1
RELATIVE POS	1-4	5-13	14-29	30-34

SCRR FACIL DESCRIPTION. The purpose of the RAMIS procedures SCRR FACIL is to determine facility requirements in hours by course and by facility. These data are stored in the RAMIS facility file FACIL.

SCRR FACIL retrieves the required hours from FACIL by course and by building and room number. Output is written on FILE FACIL. The record format for FILE FACIL is shown below:

FIELD NAME	CDP NO.	BLDG.	ROOM	REQ'D HRS.
TYPE	A	A	A	F
LENGTH	4	4	4	5.1
RELATIVE POS	1-4	5-8	9-12	13-18

SCRR FAVAIL DESCRIPTION. The purpose of the RAMIS procedure SCRR FAVAIL is to determine facility availability. Annual availability is stored in the RAMIS facility file FACIL by building and room.

A building and room combination may appear in the FACIL file more than once. Annual available hours is included on each FACIL record. SCRR FAVAIL retrieves the available hours associated with only the first appearance of each facility in the file. Output is written on FILE FAVAIL. The record format for FILE FAVAIL is shown below:

FIELD NAME	BLDG.	ROOM	AVAIL HRS.
TYPE	A	A	I
LENGTH	4	4	4
RELATIVE POS	1-4	5-8	9-12

SCRR IAVAIL DESCRIPTION. The purpose of the RAMIS procedure SCRR IAVAIL is to determine instructor availability for classroom instruction. The basic information from which availability is determined is stored in the RAMIS instructor file INST.

The instructor activity profile consists of 14 activity classifications. An estimate of the number of hours per year expended on activities within each activity classification is stored in the RAMIS instructor file for each instructor. One of the 14 activity fields (CONT) is devoted to instructor contact time and contains an annual estimate of instructor contact time. The remaining 13 fields contain annual estimates of time expended on non-contact time activities. SCRR IAVAIL sums the non-contact activities for each instructor and writes total contact time and non-contact time on FILE IAVAIL for each instructor identified

by ID and name. If the field CONT contains a Ø, a default value of one (1) hour is entered on FILE IAVAIL. The record format for FILE IAVAIL is shown below:

FIELD NAME	INST ID	INST NAME	CONT HR.	NON-CONT HR.
TYPE	I	A	I	I
LENGTH	9	16	4	4
RELATIVE POS	1-9	10-25	26-29	30-33

SCRR CRSE DESCRIPTION. The purpose of the RAMIS procedure SCRR CRSE is to record the convening frequency and average class size for each course. The required data are stored in the RAMIS course file COURSE.

SCRR CRSE first calculates average class size by multiplying class capacity by percent utilization. Convening frequency and class size are recorded by course. Output is written on FILE CRSE. The record format for FILE CRSE is shown below:

FIELD NAME	CDP NO.	CLASS SIZE	CONV
TYPE	A	I	I
LENGTH	4	3	1
RELATIVE POS	1-4	5-7	8-10

RAMSCRR EXEC FLOWCHART

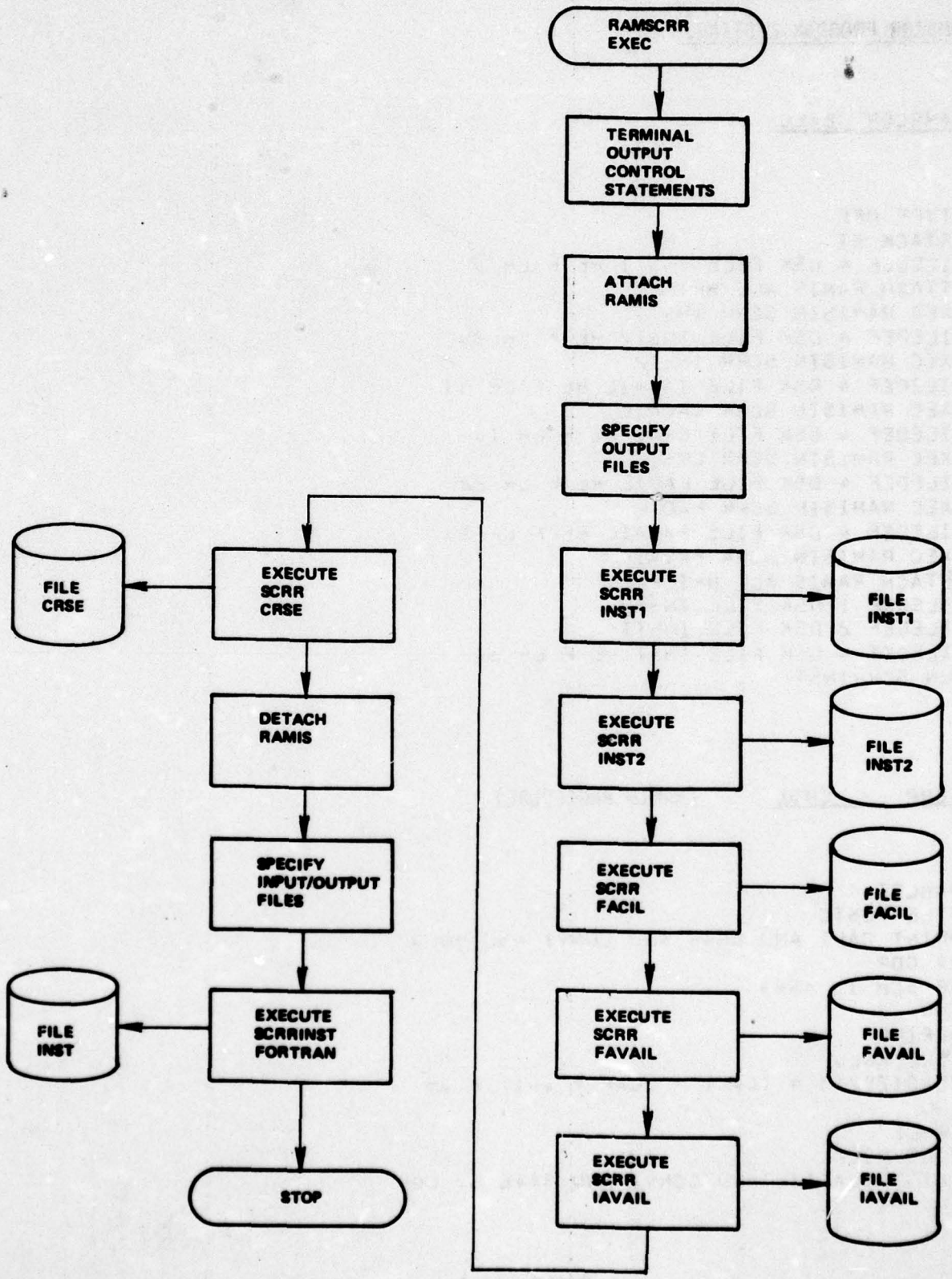


FIGURE VII-2

RAMSCRR PROGRAM LISTINGS

RAMSCRR EXEC

```
&TYPE OFF
&STACK KT
FILEDEF 4 DSK FILE INST1 RE F LR 9
ATTACH RAMIS ALL BRIEF
EXEC RAMISIN SCRR INST1
FILEDEF 4 DSK FILE INST2 RE F LR 24
EXEC RAMISIN SCRR INST2
FILEDEF 4 DSK FILE IAVAIL RE F LR 33
EXEC RAMISIN SCRR IAVAIL
FILEDEF 4 DSK FILE CRSE RE F LR 10
EXEC RAMISIN SCRR CRSE
FILEDEF 4 DSK FILE FACIL RE F LR 22
EXEC RAMISIN SCRR FACIL
FILEDEF 4 DSK FILE FAVAIL RE F LR 13
EXEC RAMISIN SCRR FAVAIL
DETACH RAMIS ALL BRIEF
FILEDEF 1 DSK FILE INST2
FILEDEF 2 DSK FILE INST1
FILEDEF 9 DSK FILE INST RE F LR 34
RUN SCRRINST
```

SCRR CRSE (RAMIS PROCEDURE)

```
TABLEF
FILE TESTC
PRINT CAP1 AND UCAP AND CONV1 AND HOLD
BY CDP
IF SCH IS ASW6
END
DEFINE
FILE HOLD
CLASIZE/I3 = (CAP1 * UCAP * .01) + .5
END
TABLE
FILE HOLD
PRINT CLASIZE AND CONV1 AND SAVE BY CDP
END
```

TABLE VII-4

SCRR FACIL (RAMIS PROCEDURE)

```
TABLE
FILE TESTF
PRINT REQHRS AND SAVE BY CDP BY BLDG BY RM
IF SCH IS ASW$
END
```

SCRR FAVAIL (RAMIS PROCEDURE)

```
TABLE
FILE TESTF
SUM FST*AVHRS AND SAVE BY BLDG BY RM
IF SCH IS ASW$
END
```

SCRR IAVAIL (RAMIS PROCEDURE)

```
TABLE
FILE TESTI
PRINT CONT AND CDEV AND EQMT AND PREP AND BKIN AND CTRN AND FTRN
AND MDUT AND ISCH AND FLET AND ADMN AND ANLV AND SKLV AND SLIB
AND HOLD BY ID BY NME
IF SCH IS ASW$
END
DEFINE
FILE HOLD
NCONT/I4= CDEV + EQMT + PREP + BKIN + CTRN + FTRN
+ MDUT + ISCH + FLET + ADMN + ANLV + SKLV + SLIB
CONTHR/I4=MAX(1,CONT)
END
TABLE
FILE HOLD
PRINT CONTHR AND NCONT AND SAVE BY ID BY NME
END
```

TABLE VII-4 (Continued)

SCRR INSI1 (RAMIS PROCEDURE)

```
TABLE
FILE TESTC
PRINT CAP1 AND UCAP AND F1 AND F2 AND F3 AND HOLD BY CDP
IF SCH IS ASW1
END
DEFINE
FILE HOLD
IF F1 GT 150 GOTO INST
AVGCLAS / I3 = (CAP1 * UCAP * .01) + .5
NOINST /I3 = MAX (1.0, (AVGCLAS / F1 ) + .5)
TOTHRS /I4 = NOINST *F2;
FLAG /I1 = 1;
GOTO END;
INST;
ASN /I1 = F3
FLAG /I1 = 0
END
TABLE
FILE HOLD
STORE TOTHRS BY CDP
IF FLAG IS 1
RUN
FILE HOLD
SUM ASN AND HOLD BY CDP IF FLAG IS 0
END
DEFINE
FILE HOLD
INSTHRS /F5.1 = TOTHRS / ASN
END
TABLE
FILE HOLD
PRINT INSTHRS AND SAVE BY CDP
END
```

TABLE VII-4 (Continued)

SCRR INST2 (RAMIS PROCEDURE)

```
TABLE
FILE TESTC
PRINT F1 AND F2 AND F3 AND NME AND HOLD BY CDP
IF SCH IS ASW
END
DEFINE
FILE HOLD
IF F1 LE 150 GOTO CRS;
ID /I9 = F1
ASN /I1 = F3
FLAG /I1 = 0;
GOTO END;
CRS;
FLAG /I1 = 1
END
TABLE
FILE HOLD
PRINT ID AND NME AND SAVE BY CDP IF ASN IS 1
IF FLAG IS 0
END
```

SCRRINST FORTRAN

```
C       COMBINES ID AND COURSE HR DATA
C       FILE 1 = FILE INST2
C       FILE 2 = FILE INST1
C       FILE 9 = FILE INST
        REAL*8 A(5),B(2)
        READ(1,50)A
 50     FORMAT (A4,A8,A1,A8,A8)
 10     READ(2,60,END=25) B
 60     FORMAT(A4,F5.1)
 15     IF (A(1) .EQ. B(1)) GO TO 20
        GO TO 10
 20     WRITE (9,70) A,B(2)
 70     FORMAT(A4,A8,A1,A8,A8,F5.1)
        READ(1,50,END=25) A
        GO TO 15
 25     STOP
        END
```

TABLE VII-4 (Continued)

BLANK EXEC DESCRIPTION

MPS III linear programming software requires that variable names should be eight characters or less in length and also should not contain embedded blanks. Building and room numbers are stored in the data base as two separate fields - a four character building name and a four character room name. Concatenation of these two names will result in a variable containing embedded blanks if the building name is less than four characters.

The purpose of the executive BLANK is to delete all embedded blank characters in the eight character facility designation contained in the files FILE FACIL and FILE FAVAIL.

Initial statements of BLANK EXEC supress all terminal output and specify libraries to be searched upon execution of subsequent PLI load commands. The input file FAVAIL for the PLI program PGM94B is identified as FILE FAVAIL and the PLI program PGM94B is executed. Then the input file FACIL for the PLI program PGM94A is identified as FILE FACIL and PGM94A is executed.

PGM94B PLI DESCRIPTION. The input file is FILE FAVAIL. The first eight characters of each thirteen character record are the building number and the room number. PGM94B reads each record and determines the index (I) of the first blank occurring in the building number. The first I-1 characters of the record are then substituted for the building number in the eight character building/room designation. The room number from the input record is then concatentated to the I-1 character building number in the output record. The output record which now contains no embedded blanks is then substituted for the input record in FILE FAVAIL.

PGM94A PLI DESCRIPTION. The only difference between PGM94A and PGM94B is the name of the input file and the input file record format. PGM94A removes embedded blanks from the eight character building/room designator located in position five of each 22 character record in FILE FACIL.

BLANK EXEC FLOWCHART

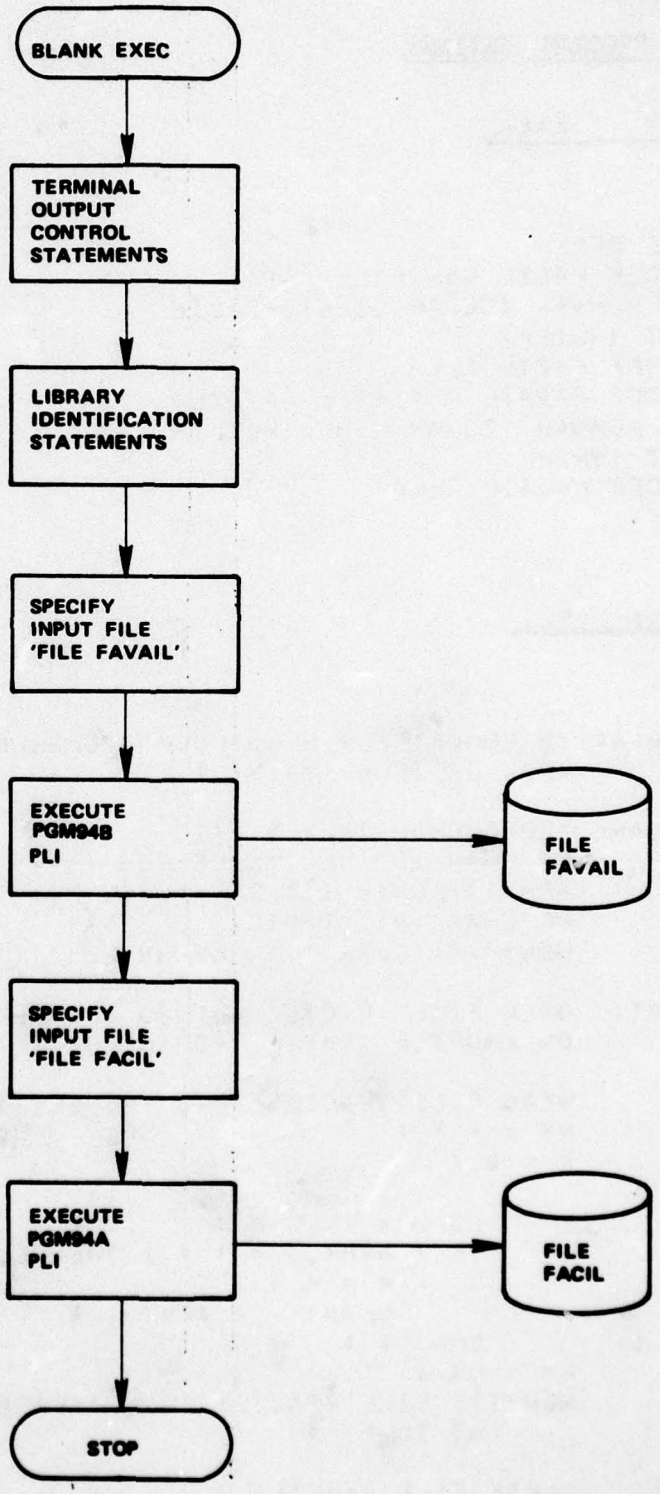


FIGURE VII-4

BLANK PROGRAM LISTINGS

BLANK EXEC

```
&TYPE OFF
FILEDEF FACIL DSK FILE FACIL
LOAD PGM94A (CLEAR LIBE) PLILIB
START (BRIEF)
FILEDEF FACIL CLEAR
FILEDEF FAVAIL DSK FILE FAVAIL
LOAD PGM94B (CLEAR LIBE) PLILIB
START (BRIEF)
FILEDEF FAVAIL CLEAR
&EXIT
```

PGM94A PLI

```
PGM94A: /* CONCAT FOR BLDG-ROOM 8 CHAR NO BLANK -2/1:/76 */
        PROC OPTIONS (MAIN) ;

DECLARE YEREC CHAR (22) STATIC,
        FB CHAR (8) DEF YEREC POS (5),
        AFB (8) CHAR (1) DEF YEREC POS (5),
        NB CHAR (8) STATIC,
        NEWB (8) CHAR (1) DEF NB POS (1) ;

START:  OPEN FILE (FACIL) RECORD UPDATE ;
        ON ENDFILE (FACIL) GO TO EOJ ;

R1:     READ FILE (FACIL) INTO (YEREC) ;
        NB = ' ' ;
        K = 0 ;

L1:         DO J = 1 TO 8 ;
            IF ( AFB(J) = ' ' ) THEN GO TO ENDL1 ;
                K = K + 1 ;
                NEWB(K) = AFB(J) ;
ENDL1:      END L1 ;
        FB = NB ;
        REWRITE FILE (FACIL) FROM (YEREC) ;
            GO TO R1 ;

EOJ:    CLOSE FILE (FACIL) ;
        END PGM94A ;
```

TABLE VII-5

PGM94B PL1

```
PGM94B: /* CONCAT FOR BLDG-ROOM 8 CHAR NO BLANK -2/1:/76 */
        PROC OPTIONS (MAIN) ;

DECLARE YEREC CHAR (13) STATIC,
        F8 CHAR (8) DEF YEREC POS (1),
        AF8 (8) CHAR (1) DEF YEREC POS (1),
        N8 CHAR (8) STATIC,
        NEW8 (8) CHAR (1) DEF N8 POS (1) ;

START:  OPEN FILE (FAVAIL) RECORD UPDATE ;
        ON ENDFILE (FAVAIL) GO TO EOJ ;

R1:     READ FILE (FAVAIL) INTO (YEREC) ;
        N8 = ' ' ;
        K = 0 ;

L1:         DO J = 1 TO 8 ;
            IF ( AF8(J) = ' ' ) THEN GO TO ENDL1 ;
                K = K + 1 ;
                NEW8(K) = AF8(J) ;
ENDL1:      END L1 ;
        F8 = N8 ;
        REWRITE FILE (FAVAIL) FROM (YEREC) ;
            GO TO R1 ;

EOJ:    CLOSE FILE (FAVAIL) ;
        END PGM94B ;
```

TABLE VII-5 (CONTINUED)

INTERFACE (IF1, IF2, IF3, and IF4) EXECS OVERVIEW

The purpose of the four interface executives IF1, IF1, IF3, and IF4 is to create a card image input data file to be processed by the MPS III linear programming routines. The linear programming input data file consists of data cards that are grouped according to data types or function. Each such group is called a segment. A segment is introduced and identified in the file by an indicator card containing the segment name. Data cards containing the data for the segment follow the indicator card.

The first card in the input data file is a NAME card containing the file name ..SCRR... The file ..SCRR.. contains ROWS, COLUMNS, RHS and BOUNDS segments. The purpose of each segment is defined below:

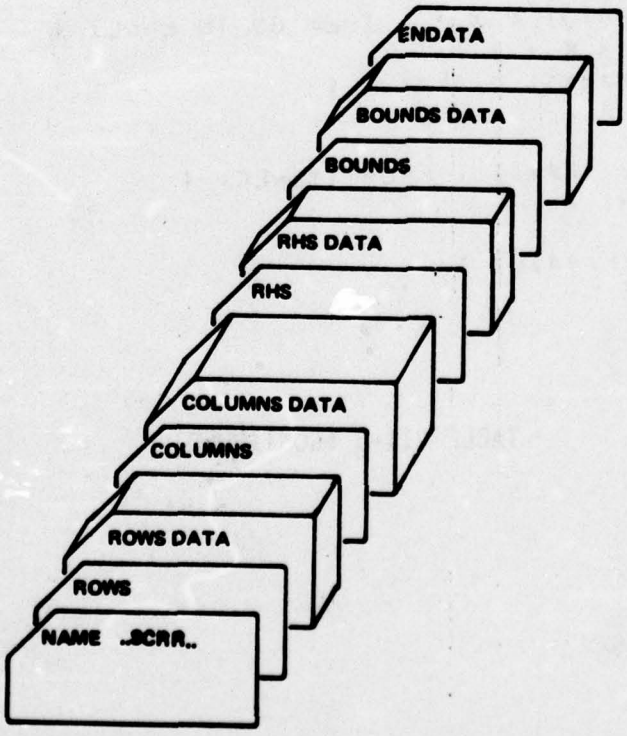
A. ROWS Segment: defines row names and types. The NAME card and the ROWS segment is generated by IF1 EXEC.

B. COLUMNS Segment: defines column names and matrix coefficients. The COLUMNS segment is generated by IF2 EXEC.

C. RHS Segment: defines right-hand-side names and coefficients. The RHS segment is generated by IF3 EXEC.

D. BOUNDS Segment: defines the column bounds set. The BOUNDS segment is generated by IF4 EXEC.

The organization of the LP data input file FTCNORVA is shown below.



DATA BASE INTERFACE EXEC FLOWCHART

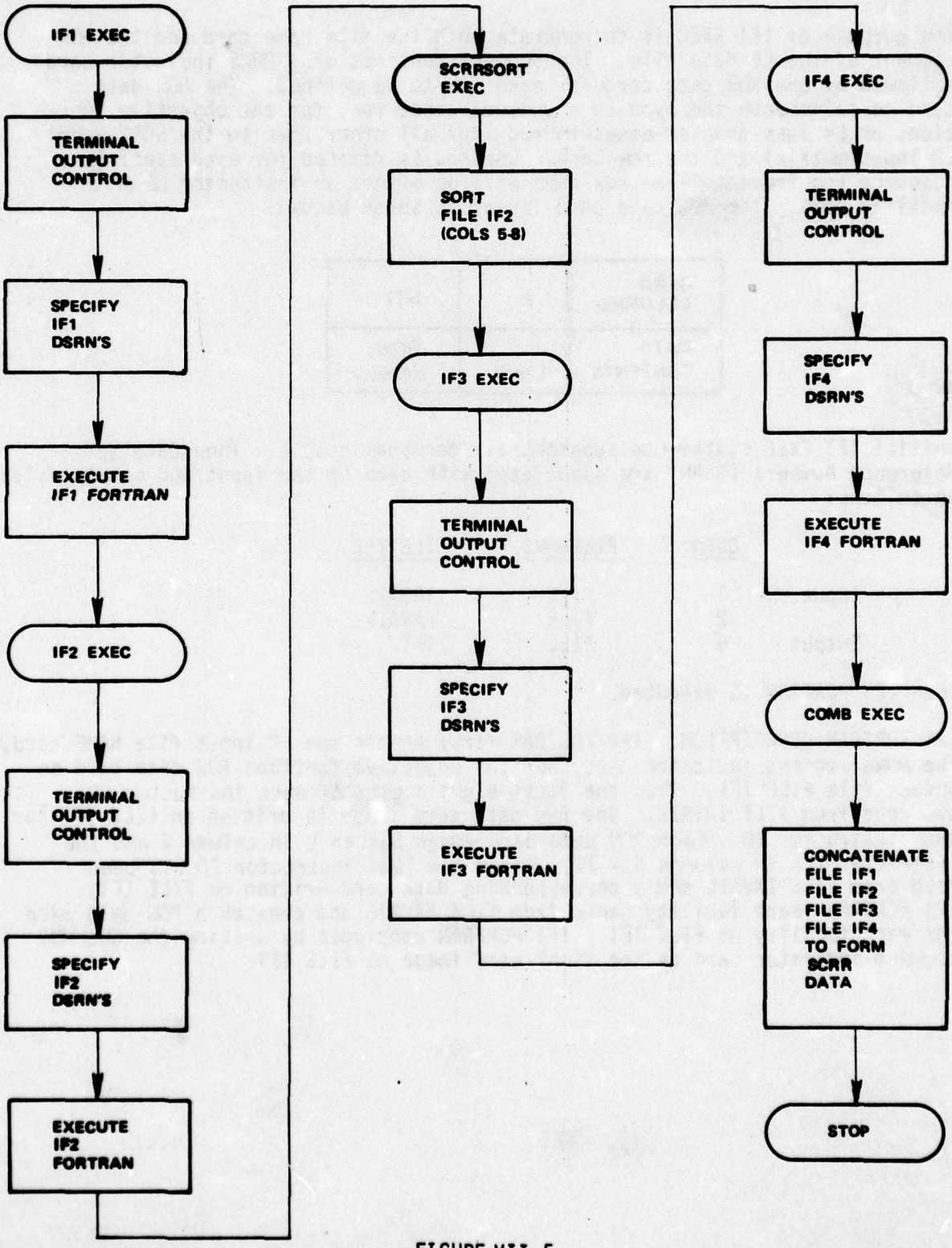


FIGURE VII-5

IF1 EXEC DESCRIPTION

The purpose of IF1 EXEC is to generate both the file name card and the ROW segment of the LP data file. The segment consists of a ROWS indicator card followed by one ROW data card for each row to be defined. The ROW data card contains both the type (N = unconstrained row, for the objective function; or L= less-than-or-equal-to row, for all other rows in the SCRR model LP input matrix) and the row name. One row is defined for each specific resource requirement. The row name will be either an instructor ID or a facility name. The ROW data card format is shown below:

CARD COLUMNS	2	5-12
DATA CONTENTS	L or N	ROW NAME

Initial IF1 EXEC statements suppress all terminal output. Then Data Set Reference Numbers (DSRN) are associated with each of the input and output files as follows:

	DSRN	FILENAME	FILETYPE
Input	1	FILE	IAVAIL
	2	FILE	FAVAIL
Output	6	FILE	IF1

Then IF1 FORTRAN IS executed.

IF1 FORTRAN DESCRIPTION. IF1 FORTRAN first writes the LP input file NAME card, the ROWS segment indicator card, and the objective function ROW data card on output file FILE IF1. Then the first eight digits of each instructor ID are read from FILE IAVAIL. One ROW data card image is written on FILE IF1 for each instructor ID. Each ROW data card image has an L in column 2 and the instructor IDs in columns 5 - 12. After the last instructor ID has been read from FILE IAVAIL and a corresponding data card written on FILE IF1, IF1 FORTRAN reads facility names from FILE FAVAIL and creates a ROW data card for each facility on FILE IF1. IF1 FORTRAN concludes by writing the COLUMNS segment indicator card as the final card image on FILE IF1.

IF1 PROGRAM LISTINGS

IF1 EXEC

```
&TYPE OFF
&STACK KT
FILEDEF 1 DSK FILE IAVAIL
FILEDEF 2 DSK FILE FAVAIL
FILEDEF 6 DSK FILE IF1 ME F LR 80
RUN IF1
```

IF1 FORTRAN

```
      REAL*8 A(2)
C
C**GENERATE NAME & ROW DATA FOR LP INPUT
C
      WRITE (6,99)
 99   FORMAT ('NAME',10X,'..SCRR..'/'ROWS'/' N  THRUPUT')
 5    READ (1,100,END = 10) A
 100  FORMAT (A8,17X,I4,4X)
      WRITE (6,101) A(1)
 101  FORMAT (' L  ',A8)
      GO TO 5
 10   READ (2,102,END = 15) A
 102  FORMAT (A8,I5)
      WRITE (6,103) A(1)
 103  FORMAT (' L  ',A8)
      GO TO 10
 15   WRITE (6,104)
 104  FORMAT ('COLUMNS')
      STOP
      END
```

TABLE VII-6

IF2 EXEC DESCRIPTION.

The purpose of IF2 EXEC is to generate the COLUMN segment of the LP data file. The COLUMN segment consists of a COLUMNS indicator card followed by at least one COLUMN data card image for each course in the RAMIS course file COURSE. Each COLUMN data card image must contain the column name (course CDP number), the row name (instructor ID or facility name, and the matrix element associated with that column and row. A second row name and matrix element on the same COLUMN card is optional. The COLUMN data card format is shown below:

CARD COLUMNS	5-12	15-22	23-36	40-47	50-61
DATA CONTENTS	COLUMN NAME	ROW NAME	MATRIX ELEMENT	ROW NAME	MATRIX ELEMENT
				OPTIONAL	

Initial IF2 EXEC statements suppress all terminal output. Then Data Set Reference Numbers (DSRN) are associated with each of the input and output files as follows:

	DSRN	FILENAME	FILETYPE
Input	1	FILE	INST
	2	FILE	FACIL
	3	FILE	CRSE
Output	6	FILE	IF2

Then IF2 FORTRAN is executed.

IF2 FORTRAN DESCRIPTION. IF2 FORTRAN formats and writes the COLUMN segment of the LP data file on file IF2 using the data contained in the three files FILE INST, FILE FACIL, and FILE CRSE. IF2 minimizes the size of the COLUMN segment by writing two row names and matrix elements on each COLUMN data card image written for columns which have two or more matrix elements.

After initializing a five column array and several indexing variables, the CDP number, instructor ID and contact hours are read from FILE INST. The CDP number is compared with the last CDP number entered into column one of the array.

If the CDP numbers do not match, the CDP number, instructor ID and contact hours are written in the first three columns of the next array row. If the CDP numbers match, the contents of column four are examined. If column four contains a zero, then the instructor ID and contact hours are written in columns four and five. If column four is not zero, then the CDP number, instructor ID and contact hours are written in the first three columns of the next row. After the last record from FILE INST has been read and entered into the array, the array contents are formatted and written on FILE IF2.

The array and indexing variables are then reset and the above procedure is repeated for FILE FACIL. After the last record of FILE FACIL has been read, the array contents are again formatted and written on FILE IF2.

FILE CRSE contains only one record for each CDP number. IF2 FORTRAN reads FILE CRSE, then formats and writes the CDP number and class size on FILE IF2.

IF2 PROGRAM LISTINGS

IF2 EXEC

```
&TYPE OFF
&STACK KT
FILEDEF 1 DSK FILE INST
FILEDEF 2 DSK FILE FACIL
FILEDEF 3 DSK FILE CASE
FILEDEF 6 DSK FILE IF2 RECFM F LRECL 80
RUN IF2
```

IF2 FORTRAN

```
      DIMENSION Z(3)
      REAL*8 A(3),M(300,5)/1500*0.0/
C**READ AND FORMAT FILE INST
C**ADD INST REQUIREMENTS TO LP COL DATA
      IRC = 0
      IR = 1
      ASSIGN 5 TO IRR
 5    READ(1,200,END = 100) A
 200  FORMAT (A4,A8,17X,F5.1)
 10   IF (A(1) .EQ. M(IR,1)) GO TO 30
      IF (M(IR,1) .NE. 0.0) IR = IR + 1
 15   IRC=IRC + 1
      DO 20  I=1,3
 20   M(IR,I) = A(I)
      GO TO IRR, (5,320)
 25   IR = IR + 1
      GO TO 15
 30   IF (M(IR,4) .NE. 0.0) GO TO 25
      DO 35  I=2,3
 35   M(IR,I+2) = A(I)
      IR = IR + 1
      GO TO IRR, (5,320)
 100  IND = 1
 110  IF (M(IND,4) .EQ. 0.0) GO TO 120
      WRITE (6,201) (M(IND,I),I=1,5)
 201  FORMAT (4X,A4,6X,A4,9X,F5.1,3X,A8,9X,F5.1)
 115  IND = IND + 1
      IF (IND .GT. IRC ) GO TO 300
      GO TO 110
 120  WRITE (6,202) (M(IND,I), I =1,3)
 202  FORMAT (4X,A4,6X,A4,9X,F5.1,25X)
      GO TO 115
```

TABLE VII-7

IF2 FORTRAN (Continued)

```
C**READ AND FORMAT FILE FACIL
C**ADD FACIL REQUIREMENTS TO LP COL DATA
 300   IRC = 0
       IR = 1
       ASSIGN 320 TO IRP
       DO 310 I=1,300
       DO 310 J=1,5
 310   M(I,J) = 0.0
 320   READ (2,203,END = 350) A
 203   FORMAT (A4,A8,F6.1,4X)
       GO TO 10
 350   IND = 1
 360   IF (M(IND,4) .EQ. 0.0) GO TO 370
       WRITE (6,204) (M(IND,I),I=1,5)
 204   FORMAT (4X,A4,6X,A8,8X,F6.1,3X,A8,8X,F6.1)
 365   IND = IND +1
       IF (IND .GT. IRC) GO TO 400
       GO TO 360
 370   WRITE (6,205) (M(IND,I),I=1,3)
 205   FORMAT (4X,A4,6X,A8,8X,F6.1,25X)
       GO TO 365
C**READ AND FORMAT FILE CRSE
C**ADD OBJ FUNC COEFF TO LP COL DATA
 400   READ (3,206,END = 405) Z
 206   FORMAT (A4,I3,I3)
       WRITE (6,207) (Z(I),I=1,2)
 207   FORMAT (4X,A4,6X,'THRUPUT',9X,I3)
       GO TO 400
 405   STOP
       END
```

TABLE VII-7 (Continued)

SCRRSORT EXEC DESCRIPTION

IF2 FORTRAN first writes instructor data by course, then facility data by course, and finally course data by course on FILE IF2. The purpose of SCRRSORT is to sort FILE IF2 by course thereby combining all records associated with each course on FILE IF2.

SCRRSORT EXEC utilizes the NCSS SORT command which moves the records from FILE IF2 to .FILE IF2 arranging them in ascending order according to sort key fields 5 - 8 which contain the course CDP number. The filename .FILE is then changed to FILE using the NCSS ALTER command.

SCRRSORT PROGRAM LISTING

SCRRSORT EXEC

```
&TYPE OFF
&STACK 5 8
SORT FILE IF2 (BRIEF)
ERASE FILE IF2
ALTER .FILE IF2 P FILE * *
```

TABLE VII-8

IF3 EXEC DESCRIPTION

The purpose of IF3 EXEC is to generate the RHS segment of the LP data file. The RHS segment consists of a RHS indicator card followed by RHS data card images which contain the RHS vector name and RHS elements. RHS elements are identified by row name (instructor ID or facility name). The RHS data card format is the same as the COLUMN data card format.

Initial IF3 EXEC statements suppress all terminal output. Then DSRN's are specified for each of the input and output files as follows:

	DSRN	FILENAME	FILETYPE
Input	1	FILE	IAVAIL
	2	FILE	FAVAIL
Output	6	FILE	IF3

Then IF3 FORTRAN is executed.

IF3 FORTRAN DESCRIPTION. IF3 FORTRAN first writes the RHS indicator card on the output file FILE IF3. Then two records each containing an instructor ID and available hours are read from FILE IAVAIL and are formatted to form a single RHS data card, which is written on FILE IF3. After the last record is read from FILE IAVAIL, two records each containing a facility name and available hours are read from FILE FAVAIL and are formatted to form a single RHS data card, which is also written on FILE IF3. The process is repeated until all records are read from FILE FAVAIL.

IF3 PROGRAM LISTINGS

IF3 EXEC

```
&TYPE OFF
&STACK KT
FILEDEF 1 DSK FILE IAVAIL
FILEDEF 2 DSK FILE FAVAIL
FILEDEF 6 DSK FILE IF3 RE F LR 80
RUN IF3
```

IF3 FORTRAN

```
      REAL*8 A,B
C
C**GENERATE RHS DATA SECTION
C  FILE 1 = FILE IAVAIL
C  FILE 2 = FILE FAVAIL
C  FILE 6 = FILE IF3
C
      WRITE (6,99)
 99   FORMAT ('RHS')
 5    READ (1,100,END = 15) A,IA
 100  FORMAT (A8,17X,I4,4X)
      READ (1,100,END = 10) B,IB
      WRITE (6,101) A,IA,B,IB
 101  FORMAT (4X,'AVAIL',5X,A8,8X,I4,5X,A8,8X,I4)
      GO TO 5
 10   WRITE (6,102) A,IA
 102  FORMAT (4X,'AVAIL',5X,A8,8X,I4)
 15   READ (2,103,END = 25) A,IA
 103  FORMAT (A8,I5)
      READ (2,103, END = 20) B,IB
      WRITE (6,101) A,IA,B,IB
      GO TO 15
 20   WRITE (6,102) A,IA
 25   STOP
      END
```

TABLE VII-9

IF4 EXEC DESCRIPTION

The purpose of IF4 EXEC is to generate the BOUNDS segment of the LP data input. The BOUNDS segment consists of a BOUNDS indicator card followed by BOUNDS data card images that contain lower bounds on the number of annual course convenings for each course. The upper bound is automatically set to +∞. The BOUNDS data card format is shown below:

CARD COLUMN	2-3	5-9	15-22	25-36
DATA CONTENTS	LO	SCHED	COLUMN NAME	BOUND VALUE

Initial IF4 EXEC statements suppress all terminal outout. Then DSRN's are specified for the input and output files as follows:

	DSRN	FILENAME	FILETYPE
Input	1	FILE	CRSE
Output	6	FILE	IF4

Then, IF4 FORTRAN is executed.

IF4 FORTRAN DESCRIPTION. IF4 FORTRAN first formats the BOUNDS indicator card on FILE IF4. Then the course CDP numbers and the number of annual course convenings are read from each record of FILE CRSE. The record is reformatted and written on FILE IF4. IF4 FORTRAN concludes by formatting an ENDATA card on FILE IF4.

IF4 PROGRAM LISTINGS

IF4 EXEC

```
&TYPE OFF
&STACK KT
FILEDEF 1 DSK FILE CHSE
FILEDEF 6 DSK FILE IF4 RE F LR 80
RUN IF4
```

IF4 FORTRAN

```
      DIMENSION A(2)
C
C**GENERATE BOUNDS DATA SECTION
C  FILE 1 = FILE CHSE
C  FILE 6= FILE IF4
C
      WRITE (6,99)
 99   FORMAT ('BOUNDS')
 5    READ (1,100, END = 10) A
 100  FORMAT (A4,3X,I3)
      WRITE (6,101) A
 101  FORMAT (' FX SCHED',5X,A4,13X,I3)
      GO TO 5
 10   WRITE (6,192)
 192  FORMAT ('ENDATA')
      STOP
      END
```

TABLE VII-10

COMB EXEC DESCRIPTION

The purpose of COMB EXEC is to combine the four output files created by the interface executives into a single input file suitable for processing by MPS III/OL.

COMB EXEC first uses the NCSS command ERASE to delete the existing copy of the file SCRR DATA from writable disk. Then SCRR DATA is recreated using the NCSS command COMBINE which joins the four files FILE IF1, FILE IF2, FILE IF3 and FILE IF4 to form SCRR DATA.

COMB PROGRAM LISTING

COMB EXEC

```
&TYPE OFF
ERASE SCRR DATA
COMBINE SCRR DATA P FILE IF1 P FILE IF2 P FILE IF3 P FILE IF4 P
```

TABLE VII-11

SCRR1 EXEC DESCRIPTION

The purpose of SCRR1 EXEC is to attach and initialize the MPS III/OL system and execute a series of Terminal Control Language (TCL) statements contained in the SCRR RUN file. MPS III is an advanced Mathematical Programming System (MPS) which has been designed to solve linear programming (LP) problems. The MPS III On-line System (MPS III/OL) is adpated for interactive use under VP/CSS, the National CSS, Inc. time-sharing system. TCL is similar to many computer programming languages (such as FORTRAN), but is oriented specifically to the solution of mathematical programming problems.

Upon entry, existing core size is checked. If core size is less than 320K, a message directing the use to SET CORE 320 is printed and the EXEC is terminated. If minimum core size requirements are met, SCRR1 EXEC then suppresses both offline print and terminal output commands contained within the MPS III/OL system. The file MPSOUT LISTING which contains a detailed record of the LP computational process is then erased. (MPSOUT LISTING is regenerated each time the SCRR model is executed). Output files for both the MPS III LP solution output and the SCRR model output are established by the FILEDEF commands.

The command ATTACH MPSIII accesses a VP/CSS system EXEC file that attaches the necessary disks, loads the Terminal Executor and requests information regarding the location of the scratch desk for work files, the sysin file filename, probfile names, card image filenames, and actfile names. Responses to these requests are stored in a special area of user storage called the CSS stack. The stack containing the MPS III file initialization information is generated by SCRR1 EXEC. The information is read from the stack each time the system EXEC requests file data. The file information is formed into an argument list and a START MPSIII TMLX command is issued. The system prompting messages,

MPS III, ON-LINE VERSION, READY
MPS COMMAND:

are then printed out at the terminal. At this point the RUN SCRR command, which has also been stacked, is executed. The SCRR OUTPUT file at this point refers to instructors by ID number and does not include non-classroom activity times for instructors. A PLI program, PGM94A, is executed to replace instructor ID numbers with names. Non-classroom times are added to the SCRR OUTPUT file by executing a FORTRAN program NCONT. SCRR1 EXEC concludes by printing a message which explains how to obtain a listing of the SCRR model output.

SCRR RUN DESCRIPTION. The purpose of SCRR RUN is to calculate the LP optimal solution and the post-optimal sensitivity analysis for the problem formulated and written on the file MATRIX1 DATA. SCRR RUN is a series of TCL statements which control the execution sequence of the LP problem.

TITLE ' ' AND TERM OFF. These two TCL commands are used to suppress MPS III/OL printed output at the user's terminal.

SET XDONFS = 16. This command is used to alter the value of the MPS III/OL Communications Region (CR) cell XDONFS. XDONFS originally contained the address of the standard procedure which terminates MPS III/OL prior to filing the solution matrix when a no-feasible-solution condition is encountered during optimization. Altering the address of the XDONFS procedure to RUN program statement 16 enables the solution output to be filed for both optimal and non-feasible-solution conditions.

CALL ASSIGN FT04F001 FT04F001 COMM. This statement specifies that MPS III/OL output is to be written on the FORTRAN file 4 in communications format.

CALL PREPOUT FT04F001. This statement positions the communications file FT04F001 to the beginning of the area to receive data and opens that file for output.

CALL ASSIGN FT07F001 FT07F001 CARD. This statement specifies that FORTRAN file 7 is to be written in card-image format.

ROUTE SYSPRINT MPSOUT. This command causes the detailed record of the LP computational process to be written on the file MPSOUT LISTING. All MPS III/OL error messages (when an error condition is encountered) will be stored on MPSOUT LISTING and will not be printed at the terminal during model execution.

SET XDATA = FTCNORVA, XPBNAME = SCRR1. This instruction alters the value of the CR location referenced by the name XDATA to the input data set name ..SCRR.. and alters the value of the location XPBNAME to the problem file name SCRR1. This allows the files to be referenced later by control program statements. The location of both the input data set and the problem file is required by the CONVERT and SETUP procedures.

CALL CONVERT. CONVERT provides the basic means of problem input. The procedure reads the input data, converts them to packed binary format, and writes them on the PROBFILE. The PROBFILE is a disk storage unit that contains the internal representation of the LP problem. While the problem is being generated, comprehensive error checks are made, and the statistics of the model are calculated for use by the SETUP procedure.

SET XOBJ - THRUPUT, XRHS = AVAIL. This instruction alters the value of the XOBJ CR location to the name of the LP objective function THRUPUT, and alters the value of the location XRHS to the name of the right-hand-side vector AVAIL.

CALL SETUP, BOUND = SCHED, MAX. SETUP reads the LP problem created by CONVERT on PROBFILE. The SETUP procedure then analyzes the model's statistics, types of input/output devices, and solution strategy to be used. The available memory is dynamically allocated for efficient problem solution and all work files used by the computational procedures are produced. SETUP also determines the initial LP basic solution. The SETUP parameter, BOUND, indicates the name of the BOUND vector in the original input data. The MAX parameter calls for the maximization of the objective function. The procedure SETUP writes the following output on the file MPSOUT LISTING:

a. Table of files that have been defined
b. Storage allocation tables
c. Table of statistics regarding the matrix variables.

CALL VARIFORM. VARIFORM is the main optimization procedure. It attempts to find a feasible solution, followed by an optimal solution. If the problem has no feasible solution, i.e., when requirements are larger than availability, an appropriate message is printed on MPSOUT LISTING and VARIFORM is terminated. Upon termination of VARIFORM, the control program branches to statement 16 which causes only the SOLUTION output to be written on FORTRAN file 4.

VARIFORM uses a composite algorithm and the revised simplex method. It progresses from the initial basis to the optimal basis by a series of vector interchanges; one vector is introduced into the basis and one is removed.

VARIFORM outputs on MPSOUT LISTING an iteration log which notes for each iteration number, the vector number removed from the basis, the vector number entered into the basis, the reduced cost of the incoming vector, and the sum of remaining infeasiblities or value of the objective function.

CALL SOLUTION FILE FT04F001 RSECTION, 2/4/7, CSECTION, 2/4/5/6. SOLUTION outputs in communications format a summary of the present LP problem and its current solution during the computational procedure if the solution is infeasible, or after the computational procedure if the solution is optimal. SOLUTION writes three arrays named SOLUTION, RSECTION and CSECTION on FORTRAN file 4. The SOLUTION array contains problem identification data. RSECTION array contains four data columns: row name (2), activity (4), slack activity (5), and upper limit (7). CSECTION array also contains four data columns: column name (2), activity (4), input cost (5), and lower limit (6).

CALL RANGE FILE FT04F001 RSECTION, 1/2/3/5, CSECTION, 2/3/4/5. Range is used post-optimally to generate and output an analysis of the current solution. The analysis includes:

a. The effects of changes in class size on optimum convening levels.

b. The impact on student throughput of changing the number of course convenings from optimum level, and the total range of course convenings for which the impact is valid.

c. The impact on student throughput of modfying a resource level and the interval for which this input is valid.

These data are tabulated in five arrays named RANGE, SECTION1, SECTION2, SECTION3, and SECTION4 and written on FORTRAN file 4. RANGE is a single row section containing information identifying the problem, model, and progress of the computation. SECTION1 and SECTION3 are row arrays each containing seven columns of data: row name (1), upper and lower activity range (2), unit cost associated with both upper and lower range value (3), and the limiting process associated with each range value (5). SECTION2 and SECTION4 are column arrays which include nine columns of data: seven columns are identical to those contained in the row arrays; column arrays also contain upper and lower cost values (4).

SKIP 1. This statement causes the next statement in the run file to be bypassed.

CALL PREPIN FT04F001. This statement positions the communications file FT04F001 to the start of the data and opens the file for input.

CSS ALTER SCRRFORM.....CALL USERPROC.....CSS ALTER USERPROC.....This series of statements changes the name of SCRRFORM LNKLIB to USERPROC LNKLIB, executes USERPROC, then changes the LNKLIB file name back to SCRRFORM.

SCRRFORM LNKLIB contains both the relocatable object program for SCRRFORM FORTRAN and the READCOMM library. READCOMM (Read Communications Format) transfers to the SCRRFORM FORTRAN procedures the data previously filed in MPS III Communications format.

EXIT. The EXIT command terminates the MPS III/OL run and returns the user to the VP/CSS command level. The system responds to the EXIT command by printing out "MPSIII TERMINATING".

SCRRFORM FORTRAN DESCRIPTION. The purpose of SCRRFORM FORTRAN is to read the MPS III/OL output which has been filed on FORTRAN file 4 in communications format, reformat the LP output, and write the results on file SCRR OUTPUT.

All MPS III/OL filed outputs are written in communications format, which cannot be read directly by the FORTRAN READ statement. Read Communications Format (READCOMM) subroutines are required to retrieve and transfer the data contained in arrays filed in Communications Format by MPSIII. SCRRFORM utilizes three READCOMM subroutines:

POSITN - positions a communications format file to a specified array.

ARRAY - initializes READCOMM to read the data contained in the array.

VECTOR - transfers a row of elements from the filed array to the calling program.

These subroutines act as an interface between the FORTRAN procedures under the control of the MPSIII executor and the Communications Format subroutines of MPSIII.

Upon entry, SCRRFORM initializes several variables, vectors and arrays. Then the RSECTION array is read from FORTRAN file 4 into the first four columns of the array R. After the last row vector has been read from RSECTION, SCRRFORM searches file 4 for the row-related arrays created by the MPSIII sensitivity analysis, SECTION1 and SECTION3. SECTION1 contains data for rows at their upper limit; SECTION3 contains data for rows at an intermediate level. If the current problem has an infeasible solution, the arrays will not exist, and the program branches to the resource (row) output format routine. If the sensitivity analysis has been performed, SCRRFORM first reads the data from SECTION1 array one row at a time. Row names are stored in column one of the array R and in column one of SECTION1. The row names in both arrays are in sorted sequence. Each row name from SECTION1 is compared with the row name in R until a match is found. Then columns two through seven of SECTION1 are read into columns five through ten of array R. The process is repeated until all rows from SECTION1

have been read into R. SCRRFORM then positions file 4 to the SECTION3 array. Row names are matched and SECTION3 data is transferred to the array R. After the last row has been read from SECTION3 and entered into R, each row of R will have ten columns of data associated with it. SCRRFORM then branches to the resource (row) output format routine, which writes column headings, and reformats the contents of R on output file SCRR OUTPUT.

After the last row of resource data has been formatted and written on the output file, SCRRFORM branches to the column section routines. Data from CSECTION are transferred to a twelve column array C, and data from SECTION2 and SECTION4 arrays (if the sensitivity analysis has been performed) are added to C, using an algorithm similar to that used to construct the row output data. Column headings are generated and the course (column) data output is appended to the resource data output on the file SCRR OUTPUT.

NCONT FORTRAN DESCRIPTION. The purpose of NCONT FORTRAN is to insert non-classroom hour data for each instructor in the SCRR OUTPUT file.

Prior to executing PGM93A PLI, instructors are identified in the left-hand column of the SCRR OUTPUT file by ID number. Non-classroom hours for each instructor ID are contained in FILE IAVAIL. Records from SCRR OUTPUT are read, ID numbers are matched with those in FILE AVAIL, and the corresponding hour figure is entered on the SCRR OUTPUT record.

PGM93A PLI DESCRIPTION. The purpose of PGM93A PLI is to replace instructor ID numbers in SCRR OUTPUT with names.

ID numbers are used as a resource variable name in the LP optimization because they are unique. However, because ID numbers are generally not easily associated back with a specific person, ID numbers are converted to names before the output data is presented. The file FILE IAVAIL contains both ID numbers and names. ID numbers occur in SCRR OUTPUT in the first column, as resource name, and in the eighth column as the limiting variable. Records are read from SCRR OUTPUT, ID numbers matched with names from FILE IAVAIL, and the ID number is replaced with the name on the SCRR OUTPUT record written on the output file.

SCRR1 EXEC FLOWCHART

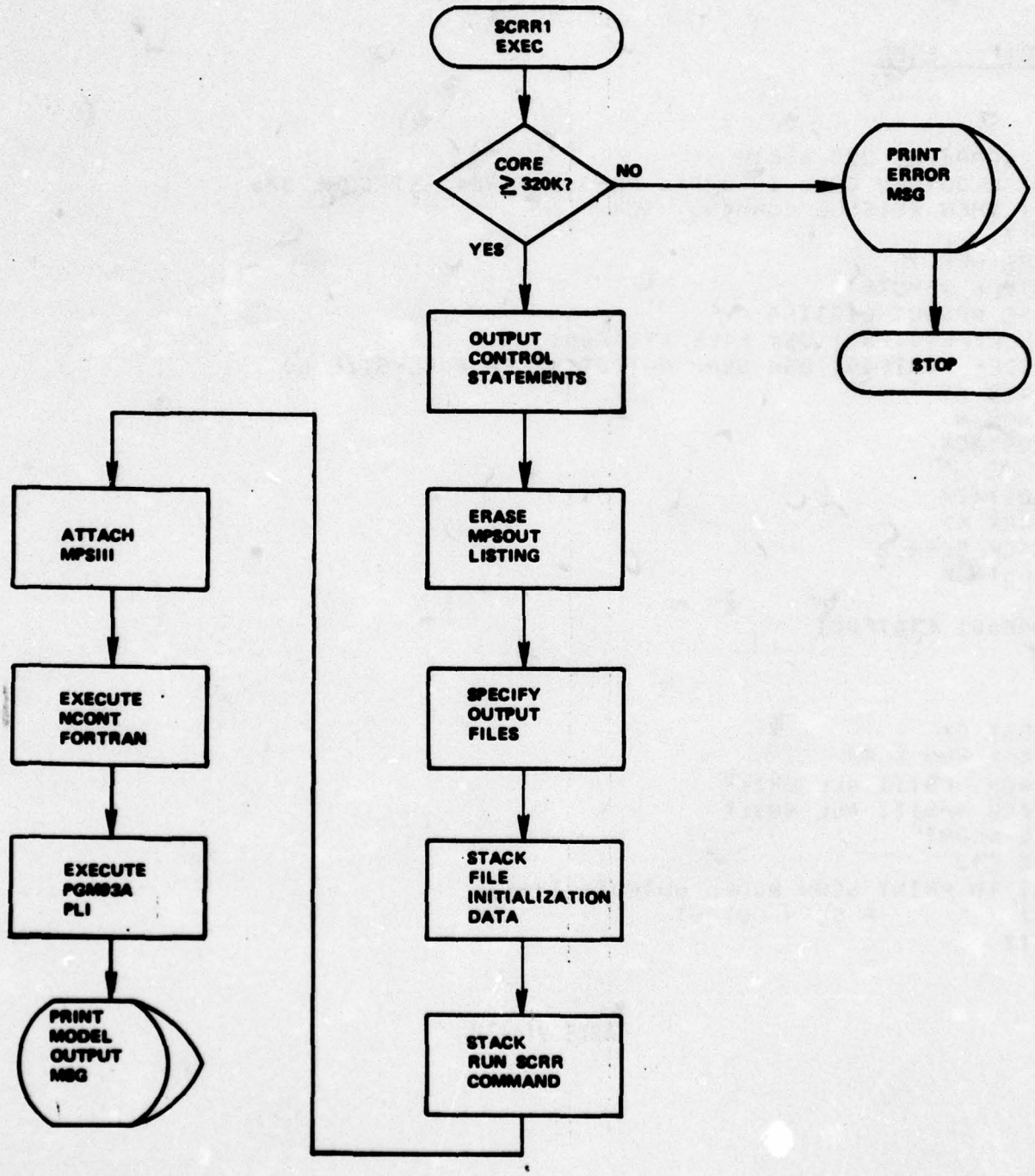


FIGURE VII-6

SCRR1 PROGRAM LISTINGS

SCRR1 EXEC

```
&IF &CORE GE 320 &SKIP 3
&PRI REQUIRED CORE IS 320K. PLEASE TYPE: SET CORE 320
&PRI THEN REISSUE COMMAND: SCRR
&EXIT
&TYPE OFF
ROUTE E REMOTE
ERASE MPSOUT LISTING
FILEDEF FT04F001 DSK FILE FT04F001
FILEDEF FT07F001 DSK SCRR OUTPUT RECFM F BLKSIZE 80
&STACK KT
&STACK N
&BEGSTACK

&ENDSTACK
&STACK KT
&STACK SCRR
&BEGSTACK

FT04F001 FT07F001


Y
&ENDSTACK
&STACK RUN SCRR
ATTACH MPSIII ALL BRIEF
DETACH MPSIII ALL BRIEF
EXEC NCONT
EXEC E93
&PRI TO PRINT SCRR MODEL OUTPUT, TYPE:
&PRI          P SCRR OUTPUT
&EXIT
```

TABLE VII-12

SCRR RUN

```
TITLE ' '
TERM OFF
SET XDONFS=16
CALL ASSIGN FT04F001 FT04F001 COMM
CALL PREPOUT FT04F001
CALL ASSIGN FT07F001 FT07F001 CARD
ROUTE SYSPRINT MPSOUT
SET XDATA=FTCNORVA,XPBNAME=SCRR1
CALL CONVERT
SET XOBJ=THROPUT,XRHS=AVAIL
CALL SETUP,ROUND=SCHED,MAX
CALL VARIFORM
CALL SOLUTION FILE FT04F001 RSECTION,2/4/5/7,CSECTION,2/4/5/6
CALL RANGE FILE FT04F001 RSECTION,1/2/3/5,CSECTION,2/3/4/5
SKIP 1
CALL SOLUTION FILE FT04F001 RSECTION,2/4/5/7,CSECTION,2/4/5/6
CALL PREPIN FT04F001
CSS ALTER SCRRFORM LNKLIB P USERPROC * *
CALL USERPROC
CSS ALTER USERPROC LNKLIB P SCRRFORM * *
EXIT
```

TABLE VII-12 (Continued)

SCRRFORM FORTRAN

```
      SUBROUTINE FORMAT
      REAL*8 A(9)/9*0.0/,NAME,R(500,10)/5000*0.0/,C(150,12)/1800*0.0/
      REAL*8 SEC1/'SECTION1'/,RSEC/'RSECTION'/,CSEC/'CSECTION'/
      REAL*8 SEC2/'SECTION2'/,SEC3/'SECTION3'/,SEC4/'SECTION4'/
      INTEGER FILE/4/
C
C GET RSECTION ARRAY
C
      CALL POSITN(FILE,I,2)
      CALL ARRAY(FILE,I,NAME,N)
      IF (I .EQ.1) GO TO 600
      IR = 1
      IRCTR= 0
 5    CALL VECTOR (FILE,I,A)
      IF (I .EQ. 1) GO TO 20
      IRCTR = IRCTR + 1
      DO 10 I=1,4
 10   R(IR,I) = A(I)
      IR = IR + 1
      GO TO 5
C
C  GET SECTION1 ARRAY
C
 20   CALL POSITN(FILE,I,5)
      CALL ARRAY(FILE,I,NAME,N)
      IF (I .EQ. 1) GO TO 200
      IF (NAME .EQ. RSEC) GO TO 200
      IF (NAME .NE. SEC1) GO TO 65
C
C ADD SENSITIVITY ANALYSIS DATA TO RESOURCE MATRIX
C
 25   IR = 1
 30   CALL VECTOR(FILE,I,A)
      IF (I .EQ. 1) GO TO 60
      DO 40 I=IR,IRCTR
      IF (A(1) .EQ. R(I,1)) GO TO 50
 40   IR = IR + 1
 50   DO 55 I = 1,6
 55   R(IR,4+I) = A(1+1)
      IR = IR + 1
      GO TO 30
 60   IF (NAME .NE. SEC1) GO TO 200
      IF (NAME .EQ. SEC3) GO TO 200
C
C GET SECTION3 ARRAY
C
 65   IA = 7
      IF (NAME .EQ. SEC1) GO TO 68
      IA = 6
 68   CALL POSITN(FILE,I,IA)
      CALL ARRAY(FILE,I,NAME,N)
      IF (I .EQ. 1) GO TO 600
      GO TO 25
C
C GET CSECTION ARRAY
```

TABLE VII-12 (Continued)

SCRRFORM FORTRAN

```
C
 70    CALL POSITN(FILE,I,3)
       CALL ARRAY(FILE,I,NAME,N)
       IF(I .EQ. 1) GO TO 600
       IR = 1
       IRCTC = 0
 75    CALL VECTOR(FILE,I,A)
       IF (I .EQ. 1) GO TO 90
       IRCTC = IRCTC + 1
       DO 80 I =1,4
 80    C(IR,I) = A(I)
       IR = IR +1
       GO TO 75
C
C GET SECTION2 ARRAY
 90    CALL POSITN(FILE,I,2)
       CALL ARRAY(FILE,I,NAME,N)
       IF (I .EQ. 1) GO TO 300
       IF (NAME .EQ. CSEC) GO TO 300
       IF (NAME .NE. SEC2) GO TO 130
C
C ADD SENSITIVITY ANALYSIS DATA TO COURSE MATRIX
C
 95    IR = 1
 100   CALL VECTOR(FILE,I,A)
       IF(I .EQ. 1) GO TO 150
       DO 110  I = IR,IRCTC
       IF (A(1) .EQ. C(I,1)) GO TO 120
 110   IR = IR + 1
 120   DO 125 I =1,8
 125   C(IR,4+I) = A(1+I)
       IR = IR + 1
       GO TO 100
 130   CALL POSITN(FILE,I,5)
       CALL ARRAY(FILE,I,NAME,N)
       IF (I .EQ. 1) GO TO 600
       IF (NAME .EQ. SEC2) GO TO 95
C
C GET SECTION4 ARRAY
C
 150   IF (NAME .EQ. SEC4) GO TO 300
       CALL POSITN(FILE,I,6)
       CALL ARRAY(FILE,I,NAME)
       IF (NAME .EQ. SEC4) GO TO 95
       IF (NAME .EQ. SEC3) GO TO 300
       CALL POSITN(FILE,I,7)
       CALL ARRAY(FILE,I,NAME)
       IF (NAME .EQ. SEC4) GO TO 95
       CALL POSITN(FILE,I,8)
       CALL ARRAY(FILE,I,NAME)
       IF (I .EQ. 1) GO TO 300
       GO TO 95
C
C FORMAT RESOURCE OUTPUT
C
```

TABLE VII-12 (Continued)

SCRRFORM FORTRAN

```
 200  WRITE(7,500)
 500  FORMAT (12X,'ANNUAL     ANNUAL     ANNUAL     REQMT     RESOURCE',
     1' THROUGHPUT'/13X,'NON-      AVAIL      REQMT',7X,'VS',6X,
     1'UTIL.      CHANGE    LIMITING'/1X,'RESOURCE',
     1' CLASSROOM CLASSROOM CLASSROOM    AVAIL      RANGE     PER UNIT  ',
     1'VARIABLE'/12X,'HOURS     CONTACT    CONTACT     RATIO     MINIMUM',
     1'    RESOURCES'/22X,'HOURS      HOURS',14X,'MAXIMUM     CHANGE')
      IR = 2
 210  I4 = R(IR,4) + 0.5
      I2 = R(IR,2) + 0.5
      I5 = (R(IR,2) / R(IR,4)) * 100
      IF ((R(IR,5).EQ.0).AND.(R(IR,6).EQ.0).AND.(R(IR,7).EQ.0))GO TO 250
      IF (R(IR,6) .LT. -1.E10) GO TO 215
      IF (R(IR,9) .LT. -1.E10) GO TO 220
      WRITE(7,501) R(IR,1),I4,I2,I5,(R(IR,I),I=5,10)
 501  FORMAT (1X,A8,2X,6X,4X,I6,4X,I6,6X,I4,3X,F8.1,3X,
     1F7.3,3X,A8/ 50X,F8.1,3X,F7.3,3X,A8)
 211  IR = IR + 1
      IF (IR .LE. IRLTR) GO TO 210
      GO TO 70
 215  IF (R(IR,9) .LT. -1.E10) GO TO 225
      WRITE(7,502) R(IR,1),I4,I2,I5,R(IR,5),(R(IR,I),I=7,10)
 502  FORMAT (1X,A8,2X,6X,2(4X,I6),6X,I4,3X,F8.1,4X,'NO CHG',3X,A8/
     150X,F8.1,3X,F7.3,3X,A8)
      GO TO 211
 220  WRITE(7,503) R(IR,1),I4,I2,I5,(R(IR,I),I=5,8),R(IR,10)
 503  FORMAT (1X,A8,2X,6X,2(4X,I6),6X,I4,3X,F8.1,3X,F7.3,3X,A8/
     150X,F8.1,4X,'NO CHG',3X,A8)
      GO TO 211
 225  WRITE (7,520) R(IR,1),I4,I2,I5,R(IR,5),R(IR,7),R(IR,8),R(IR,10)
 520  FORMAT (1X,A8,2X,6X,2(4X,I6),6X,I4,3X,F8.1,4X,'NO CHG',3X,A8/
     150X,F8.1,4X,'NO CHG',3X,A8)
      GO TO 211
 250  WRITE (7,517) R(IR,1),I4,I2,I5
 517  FORMAT (1X,A8,2X,6X,2(4X,I6),6X,I4/' ')
      GO TO 211
C
C FORMAT COURSE OUTPUT
C
 300  WRITE(7,504)
 504  FORMAT (' '/' '/' '/42X,'ANNUAL  THROUGHPUT  CLASS'/2X,'COURSE',
     1'     MAXIMUM   CURRENT   AVERAGE   CONVEN    CHANGE      SIZE  ',
     1'  LIMITING'/ 2X,'CDP NO.   ANNUAL   SCHEDULED   CLASS    RANGE',
     1'    PER COURSE  RANGE    VARIABLE'/ 12X,'CONVEN    CONVEN   ',
     1'  SIZE    MINIMUM    CONVEN   MINIMUM'/ 41X,'MAXIMUM    ',
     1'CHANGE   MAXIMUM')
      IR=1
 310  I4 = C(IR,4) + 0.5
      I3 = C(IR,3) + 0.5
      IF((C(IR,5).EQ.0).AND.(C(IR,6).EQ.0).AND.(C(IR,7).EQ.0))GO TO 400
      IF (C(IR,5) .LT. -1.E10) GO TO 350
      IF (C(IR,7) .LT. -1.E10) GO TO 340
      IF (C(IR,10).LT. -1.E10) GO TO 325
      IF (C(IR,11).GT.  1.E10) GO TO 320
      WRITE(7,505) C(IR,1),C(IR,2),I4,I3,(C(IR,I),I=5,12)
```

TABLE VII-12 (Continued)

```
505  FORMAT (3X,A4,4X,F6.1,2(6X,I4),3(4X,F6.1),4X,A8/
     137X,3(4X,F6.1),4X,A8)
315  IR=IR + 1
     IF (IR .LE. IRCIC) GO TO 310
     RETURN
320  WRITE(7,506) C(IR,1),C(IR,2),I4,I3,(C(IR,I),I=5,10),C(IR,12)
506  FORMAT (3X,A4,4X,F6.1,2(6X,I4),3(4X,F6.1),4X,A8/
     137X,2(4X,F6.1),6X,'+INF',4X,A8)
     GO TO 315
325  IF (C(IR,11) .GT. 1.E10) GO TO 330
     WRITE(7,507) C(IR,1),C(IR,2),I4,I3,(C(IR,I),I=5,9),C(IR,11),
     1C(IR,12)
507  FORMAT (3X,A4,4X,F6.1,2(6X,I4),3(4X,F6.1),4X,A8/
     137X,4X,F6.1,4X,'NO CHG',4X,F6.1,4X,A8)
     GO TO 315
330  WRITE(7,508) C(IR,1),C(IR,2),I4,I3,(C(IR,I),I=5,9),C(IR,12)
508  FORMAT (3X,A4,4X,F6.1,2(6X,I4),3(4X,F6.1),4X,A8/ 37X,4X,F6.1,
     14X,'NO CHG',6X,'+INF',4X,A8)
     GO TO 315
340  IF (C(IR,10) .LT. -1.E10) GO TO 345
     WRITE(7,509) C(IR,1),C(IR,2),I4,I3,C(IR,5),C(IR,6),
     1(C(IR,I),I=8,12)
509  FORMAT (3X,A4,4X,F6.1,2(6X,I4),2(4X,F6.1),6X,' 0.0',4X,A8/
     137X,3(4X,F6.1),4X,A8)
     GO TO 315
345  WRITE(7,510) C(IR,1),C(IR,2),I4,I3,C(IR,5),C(IR,6),C(IR,8),
     1C(IR,9),C(IR,11),C(IR,12)
510  FORMAT (3X,A4,4X,F6.1,2(6X,I4),2(4X,F6.1),6X,' 0.0',4X,A8/
     137X,4X,F6.1,6X,'+INF',4X,A8)
     GO TO 315
350  IF (C(IR,7) .LT. -1.E10) GO TO 370
     IF (C(IR,10).LT. -1.E10) GO TO 360
     IF (C(IR,11).GT.  1.E10) GO TO 355
     WRITE(7,511) C(IR,1),C(IR,2),I4,I3,(C(IR,I),I=6,12)
511  FORMAT (3X,A4,4X,F6.1,2(6X,I4),5X,' 0.0',2(4X,F6.1),4X,A8/
     137X,3(4X,F6.1),4X,A8)
     GO TO 315
355  WRITE(7,512) C(IR,1),C(IR,2),I4,I3,(C(IR,I),I=6,10),C(IR,12)
512  FORMAT (3X,A4,4X,F6.1,2(6X,I4),6X,' 0.0',2(4X,F6.1),4X,A8/
     137X,2(4X,F6.1),6X,'+INF',4X,A8)
     GO TO 315
360  IF (C(IR,11) .GT. 1.E10) GO TO 365
     WRITE(7,513) C(IR,1),C(IR,2),I4,I3,(C(IR,I),I=6,9),C(IR,11),
     1C(IR,12)
513  FORMAT (3X,A4,4X,F6.1,2(6X,I4),6X,' 0.0',2(4X,F6.1),4X,A8/
     137X,4X,F6.1,4X,'NO CHG',4X,F6.1,4X,A8)
     GO TO 315
365  WRITE(7,514) C(IR,1),C(IR,2),I4,I3,(C(IR,I),I=5,9),C(IR,12)
514  FORMAT (3X,A4,4X,F6.1,2(6X,I4),6X,' 0.0',2(4X,F6.1),4X,A8/
     137X,4X,F6.1,4X,'NO CHG',6X,'+INF',4X,A8)
     GO TO 315
370  IF (C(IR,10) .LT. -1.E10) GO TO 375
     WRITE(7,515) C(IR,1),C(IR,2),I4,I3,C(IR,6),(C(IR,I),I=8,12)
515  FORMAT (3X,A4,4X,F6.1,2(6X,I4),6X,' 0.0',4X,F6.1,6X,' 0.0',
     14X,A8/ 37X,3(4X,F6.1),4X,A8)
```

TABLE VII-12 (Continued)

SCRRFORM FORTRAN

```
      GO TO 315
 375  WRITE(7,516) C(IR,1),C(IR,2),I4,I3,C(IR,6),C(IR,8),C(IR,9),
     1C(IR,11),C(IR,12)
 516  FORMAT (3X,A4,4X,F6.1,2(6X,I4),6X,' 0.0',4X,F6.1,5X,' 0.0',
     14X,A8/ 37X,4X,F6.1,4X,'NO CHG',4X,F6.1,4X,A8)
      GO TO 315
 400  WRITE (7,518) C(IR,1),C(IR,2),I4,I3
 518  FORMAT (3X,A4,4X,F6.1,2(6X,I4)/' ')
      GO TO 315
 600  WRITE (7,519)
 519  FORMAT('***  READCOMM  END OF FILE ERROR!!!!!!!')
      END
```

NCONT FORTRAN

```
C**FILE 1 = SCRR OUTPUT
C**FILE 2 = FILE 1AVAIL
C**FILE 9 = SCRR OUTPUT(NEW)
C
      INTEGER*2 N(8),M(8),BL/' '/
      REAL*8 A(11),B
      DO 5 I=1,8
      READ(1,100) N,A
 100  FORMAT (1X,8A1,4A4,6A8,A7)
   5  WRITE (9,100) N,A
  10  READ (1,100) N,A
  15  READ (2,101,END=50) M,B
 101  FORMAT (8A1,21X,A4)
  20  IF(N(8) .EQ. BL) GOTO 55
      DO 23 I=1,8
      IF (N(I) .NE. M(I)) GOTO 15
  23  CONTINUE
  25  WRITE (9,102) N,B,(A(I),I=3,11)
 102  FORMAT (1X,8A1,4X,A4,2A4,6A8,A7)
      READ (1,100) N,A
      WRITE (9,100) N,A
      GO TO 10
  50  WRITE (9,100) N,A
      READ (1,100,END=60) N,A
      GO TO 50
  55  DO 56 I=1,8
      J=9-I
  56  N(J) = N(J-1)
      N(1) = BL
      GOTO 20
  60  STOP
      END
```

TABLE VII-12 (Continued)

PGM93A PLI

```
PGM93A: PROC OPTIONS (MAIN) ;

DCL     CARD CHAR (80) STATIC,
        F8 CHAR (8) DEF CARD POS (2),
        C80 CHAR (33) STATIC,
        F8C CHAR (8) DEF C80 POS (1),
        NAMEC CHAR (8) DEF C80 POS (10) ;

DCL     LV CHAR (8) DEF CARD POS (72),
        TEST CHAR (8) STATIC,
        (ITAB(300), NTAB(300)) CHAR (8),
        (FIND, NOFIND) LABEL,
        (FLAG, ANYLEFT) BIT (1) STATIC ;

START:  OPEN FILE (SOUT) RECORD UPDATE,
             FILE (IAVAIL) RECORD INPUT ;
        ON ENDFILE (IAVAIL) GO TO PASS2 ;
        ON ENDFILE (SOUT) GO TO EOF ;
        FLAG = '0'B ;
        ANYLEFT = '1'B ;

        NT = 0 ;
RD1:    READ FILE (IAVAIL) INTO (C80) ;
        NT = NT + 1 ;
        ITAB(NT) = F8C ;
        NTAB(NT) = NAMEC ;
             GO TO RD1 ;

PASS2:
R1:     READ FILE (SOUT) INTO (CARD) ;
        IF ( F8 ¬= 'RESOURCE' ) THEN GO TO R1 ;
R2:     READ FILE (SOUT) INTO (CARD) ;
        IF ( F8 = '        ' ) THEN GO TO R2 ;
             GO TO RUN ;

R3:     READ FILE (SOUT) INTO (CARD) ;
LEFT:   IF ( ¬ANYLEFT ) THEN GO TO RIGHT ;
        IF ( F8 = '        ' ) THEN GO TO RIGHT ;

        IF ( F8 = ' COURSE ' ) THEN
NULLIFYLEFT:   DO ;
               ANYLEFT = '0'B ;
               GO TO RIGHT ;
               END NULLIFYLEFT ;
```

TABLE VII-12 (Continued)

PGM93A PLI

```
RUN:      TEST = F8 ;
          FIND = FIXNAME ;
          NOFIND = RIGHT ;
                   GO TO LOOKSR ;

FIXNAME:  F8 = NTAB (J) ;
          FLAG = '1'B ;

RIGHT:    IF ( LV = '          ' ) THEN GO TO WINDUP ;
RUN2:     TEST = LV ;

          FIND = FIXRIGHT ;
          NOFIND = WINDUP ;
                   GO TO LOOKSR ;

FIXRIGHT: LV = NTAB (J) ;
          FLAG = '1'B ;

WINDUP:   IF ( FLAG ) THEN
                   REWRITE FILE (SOUT) FROM (CARD) ;
          FLAG = '0'B ;
                   GO TO R3 ;

LOOKSR:   J = 1 ;
LOOKLOOP: IF ( TEST = ITAB(J) ) THEN GO TO FIND ;
TALLY:    J = J + 1 ;
          IF ( J <= NT ) THEN GO TO LOOKLOOP ;
                   GO TO NOFIND ;

EOF: EOJ: CLOSE FILE (SOUT), FILE (IAVAIL) ;

          END PGM93A ;
```

TABLE VII-12 (Continued)

ASSIGN EXEC DESCRIPTION

The purpose of ASSIGN EXEC is to generate an instructor assignment listing which contains the following data for each instructor:

- o assigned courses
- o projected average class size
- o contact hour requirement per convening
- o scheduled convenings per year
- o total annual contact hour requirement per course
- o projected annual availability

Initial ASSIGN EXEC statements suppress all terminal output. Then DSRN's are associated with each of the input and output files as follows:

	DSRN	FILENAME	FILETYPE
Input	1	FILE	INST
	2	FILE	CRSE
Output	9	FILE	ASSIGN

After DSRN specification, ASSIGN1 FORTRAN is executed. ASSIGN1 FORTRAN output file FILE ASSIGN is then sorted by instructor using the NCSS SORT command. The sort key fields are 11-18. DSRN's are then reassigned as follows:

	DSRN	FILENAME	FILETYPE
Input	1	FILE	ASSIGN
	2	FILE	IAVAIL
Output	8	ASSIGN	LISTING

ASSIGN2 FORTRAN is then executed and instructions on how to obtain output are printed.

ASSIGN1 FORTRAN DESCRIPTION. The purpose of ASSIGN1 FORTRAN is to combine class size and convening data from FILE CRSE with each FILE INST record. FILE INST contains one record for each instructor assigned to a course. FILE CRSE contains one record for each course. Both files are sorted by CDP number.

ASSIGN1 reads the first record from both FILE INST and FILE CRSE. If both records contain the same CDP number, the data from both records are combined to form a single record which is written on FILE ASSIGN. The next record is then read from FILE INST. The CDP number is compared with the CDP number in the last record read from FILE CRSE. If they are identical, the data are combined and a single record is written on the output file. If the CDP numbers are not the same, the next record is read from FILE CRSE and the CDP comparison is repeated.

Records are read from both files, combined and rewritten on FILE ASSIGN until the last record is read from FILE INST.

ASSIGN2 FORTRAN DESCRIPTION. The purpose of ASSIGN2 FORTRAN is to combine instructor requirements and availability data and to format the data on the output file ASSIGN listing. Both input files FILE ASSIGN and FILE IAVAIL are sorted by instructor.

Upon entry, ASSIGN2 FORTRAN writes a heading on ASSIGN LISTING. Then the first record is read from FILE ASSIGN. Total contact hour requirements are calculated by multiplying the hours per convening by the number of convenings. The next record is read, total hours calculated, and the instructor name is compared with the instructor name of the prior record. If the names are different, availability is read from FILE IAVAIL, the former record is reformatted and written on ASSIGN LISTING, and the process is repeated.

If the names are identical, the first record is formatted and written on the output file, the next record read from FILE ASSIGN, and the instructor names are again compared. If the names are the same, the process is repeated. If the names are different, the former record is formatted and written on ASSIGN LISTING. Then availability is read from FILE IAVAIL, and a summary record, which includes total contact hours for all course assignments and total availability, is written on the output file.

Records are read from FILE ASSIGN, names compared, and output records written (format is a function of single vs. multiple assignments) until the last record is read from FILE ASSIGN.

ASSIGN EXEC FLOWCHART

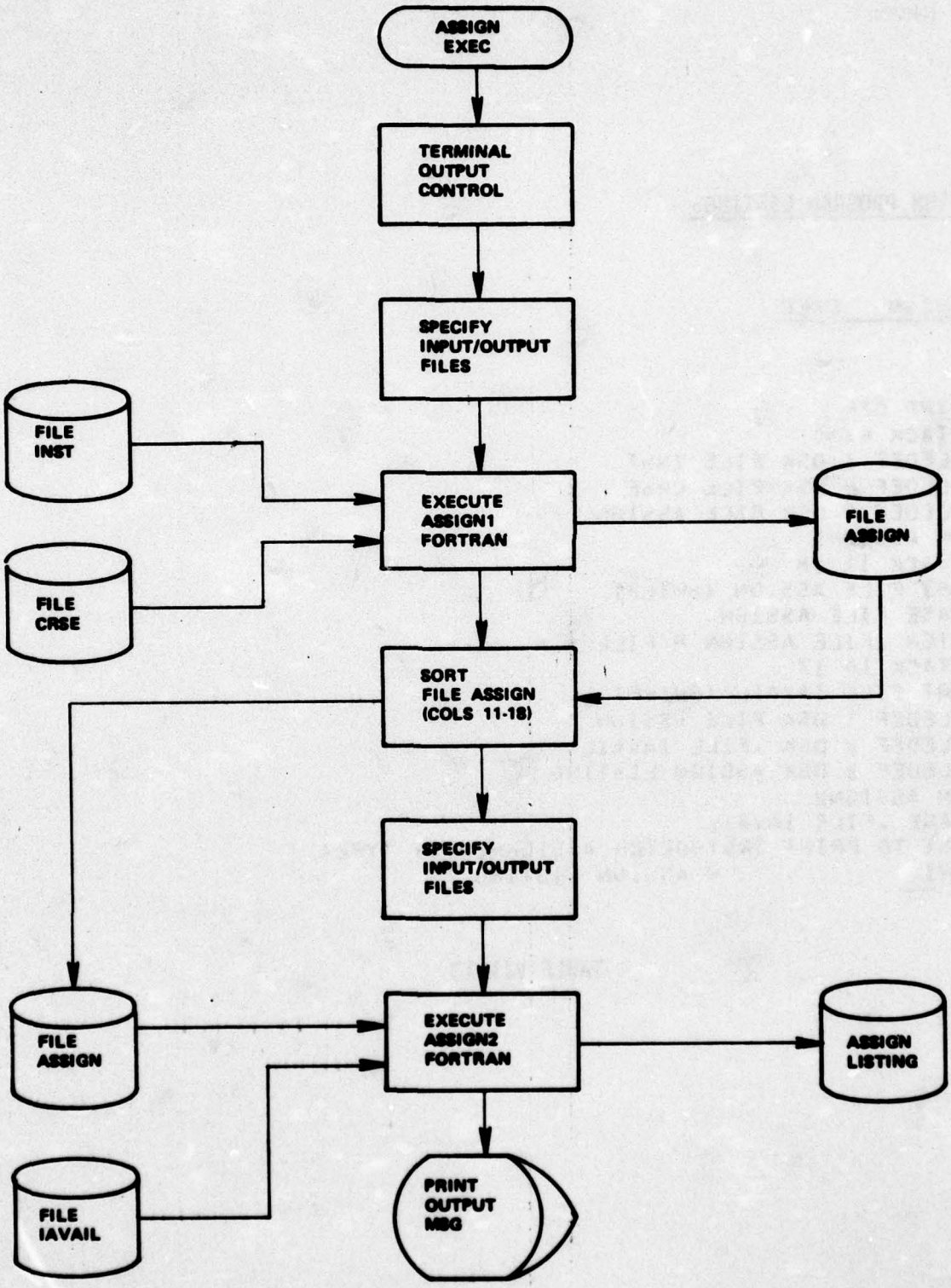


FIGURE V-II-7

ASSIGN PROGRAM LISTINGS

ASSIGN EXEC

```
&TYPE OFF
&STACK KT
FILEDEF 1 DSK FILE INST
FILEDEF 2 DSK FILE CHSE
FILEDEF 9 DSK FILE ASSIGN
RUN ASSIGN1
&STACK 11 18
SORT FILE ASSIGN (BRIEF)
ERASE FILE ASSIGN
ALTER .FILE ASSIGN P FILE * *
&STACK 10 17
SORT FILE IAVAIL (BRIEF)
FILEDEF 1 DSK FILE ASSIGN
FILEDEF 2 DSK .FILE IAVAIL
FILEDEF 6 DSK ASSIGN LISTING
RUN ASSIGN2
ERASE .FILE IAVAIL
&PRI TO PRINT INSTRUCTOR ASSIGNMENTS, TYPE:
&PRI              P ASSIGN LISTING
```

TABLE VII-13

ASSIGN1 FORTRAN

```
      REAL*8 A(3),B
C**COMBINES INST AND CRSE DATA INTO SINGLE FILE
C  FILE 1 = FILE INST
C  FILE 2 = FILE CRSE
C  FILE 9 = FILE ASSIGN
C
      READ (1,500) A
 500  FORMAT (A4,9X,A8,8X,F5.1)
 10   READ (2,501, END = 25) B,IB1,IB2
 501  FORMAT (A4,I3,I3)
 15   IF (A(1) .EQ. B ) GO TO 20
      GO TO 10
 20   WRITE (9,502) A(1),IB1,IB2,A(2),A(3)
 502  FORMAT (A4,I3,I3,A8,F5.1)
      READ (1,500, END = 25) A
      GO TO 15
 25   STOP
      END
```

ASSIGN2 FORTRAN

```
      REAL*8 A(5),B(5),C
C
C**FORMATS INST ASSIGNMENT DATA FOR OUTPUT
C  FILE 1 = FILE ASSIGN
C  FILE 2 = FILE IAVAIL
C  FILE 8 = ASSIGN LISTING
C
      WRITE (8,100)
 100  FORMAT ('  INSTRUCTOR   COURSE  CLASIZE  CONV  HRS/CONV',
     1'  TOT HRS  AVAIL HRS'/'  ----------   ------  -------  ',
     1'----  --------  -------  ---------')
      READ (1,99) A(1),IA1,IA,A(4),A(5)
 99   FORMAT (A4,I3,I3,A8,F5.1)
      A(3) = IA
      TA = A(3) * A(5)
      TOT = TA
 10   READ (1,99, END = 150) B(1),IB1,IB,B(4),B(5)
      B(3) = IB
      TB = B(3) * B(5)
      IF ( A(4) .NE. B(4) ) GO TO 50
      WRITE (8,101) A(4),A(1),IA1,IA,A(5),TA
 101  FORMAT (2X,A8,6X,A4,5X,I3,4X,I3,5X,F5.1,4X,F6.1)
```

TABLE VII-13 (Continued)

ASSIGN2 FORTRAN

```
25    TOT = TOT + TB
      DO 30  I=1,5
30    A(I) = B(I)
      IA1 = IB1
      IA = IB
      TA = TB
      READ (1,99, END = 175) B(1),IB1,IB,B(4),B(5)
      B(3) = IB
      TB = B(3) * B(5)
      IF ( A(4) .NE. B(4) ) GO TO 90
      WRITE (8,102) A(1),IA1,IA,A(5),TA
102   FORMAT (16X,A4,5X,I3,4X,I3,5X,F5.1,4X,F6.1)
      GO TO 25
50    READ (2,104) C,IC
104   FORMAT (A8,17X,I4,4X)
      WRITE (8,105) A(4),A(1),IA1,IA,A(5),TA,IC
105   FORMAT (2X,A8,6X,A4,5X,I3,4X,I3,5X,F5.1,4X,F6.1,4X,I4/)
55    DO 60  I=1,5
60    A(I) = B(I)
      IA1 = IB1
      IA = IB
      TA = TB
      TOT = TA
      GO TO 10
90    READ (2,104) C,IC
      WRITE (8,102) A(1),IA1,IA,A(5),TA
      WRITE (8,106) TOT,IC
106   FORMAT (49X,'------'/43X,'TOT = ',F6.1,4X,I4/)
      GO TO 55
150   READ (2,104) C,IC
      WRITE (8,105) A(4),A(1),IA1,IA,A(5),TA,IC
      CALL EXIT
175   READ (2,104) C,IC
      WRITE (8,102) A(1),IA1,IA,A(5),TA
      WRITE (8,106) TOT,IC
      CALL EXIT
      END
```

TABLE VII-13 (Continued)

FACILITY EXEC DESCRIPTION

The purpose of FACILITY EXEC is to generate a facility utilization listing which contains the following data for each facility:

- o assigned courses
- o projected average class size
- o contact hour requirement per convening
- o scheduled convenings per year
- o total annual contact hour requirement per course
- o projected annual availability

Initial FACILITY EXEC statements suppress all terminal output. Then DSRN's are associated with each of the input and output files as follows:

	DSRN	FILENAME	FILETYPE
Input	1	FILE	FACIL
	2	FILE	CRSE
Output	9	FILE	FAC

After DSRN specification, FAC1 FORTRAN is executed. FAC1 FORTRAN output file FILE FAC is then sorted by facility using the NCSS SORT command. The sort key fields are 11-18. DSRN's are then reassigned as follows:

	DSRN	FILENAME	FILETYPE
Input	1	FILE	FAC
	2	FILE	FAVAIL
Output	8	FACILITY	LISTING

FAC2 FORTRAN is then executed and instructions on how to obtain output are printed.

FAC1 FORTRAN DESCRIPTION. The purpose of FAC1 FORTRAN is to combine class size and convening data from FILE CRSE with each FILE FACIL record. FILE FACIL contains one record for each course/facility combination. FILE CRSE contains one record for each course. Both files are sorted by CDP number.

FAC1 reads the first record from both FILE FACIL and FILE CRSE. If both records contain the same CDP number, the data from both records are combined to form a single record which is written on FILE FAC. The next record is then read from FILE FACIL. The CDP number is compared with the CDP number in the last record read from FILE CRSE. If they are identical, the data are combined and a single record is written on the output file. If the CDP numbers are not the same, the next record is read from FILE CRSE and the CDP comparison is repeated.

Records are read from both files, combined and rewritten on FILE FAC until the last record is read from FILE FACIL.

FAC2 FORTRAN DESCRIPTION. The purpose of FAC2 FORTRAN is to combine facility requirements and availability data and to format the data on the output file FACILITY LISTING. Both input files FILE FAC and FILE FAVAIL are sorted by instructor.

Upon entry, FAC2 FORTRAN writes a heading on FACILITY LISTING. Then the first record is read from FILE FAC. Total contact hour requirements are calculated by multiplying the hours per convening by the number of convenings. The next record is read, total hours calculated, and the building/room number is compared with the building/room number of the prior record. If the numbers are different, availability is read from FILE FAVAIL, the former record is reformatted and written on FACILITY LISTING, and the process is repeated.

If the names are identical, the first record is formatted and written on the output file, the next record read from FILE FAC, and the building/room numbers are again compared. If the numbers are the same, the process is repeated. If the numbers are different, the former record is formatted and written on FACILITY LISTING. Then availability is read from FILE FAVAIL, and a summary record, which includes total contact hours for all course assignments and total availability, is written on the output file.

Records are read from FILE FAC, numbers compared, and output records written (format is a function of single vs. multiple room assignments) until the last record is read from FILE FAC.

FACILITY EXEC FLOWCHART

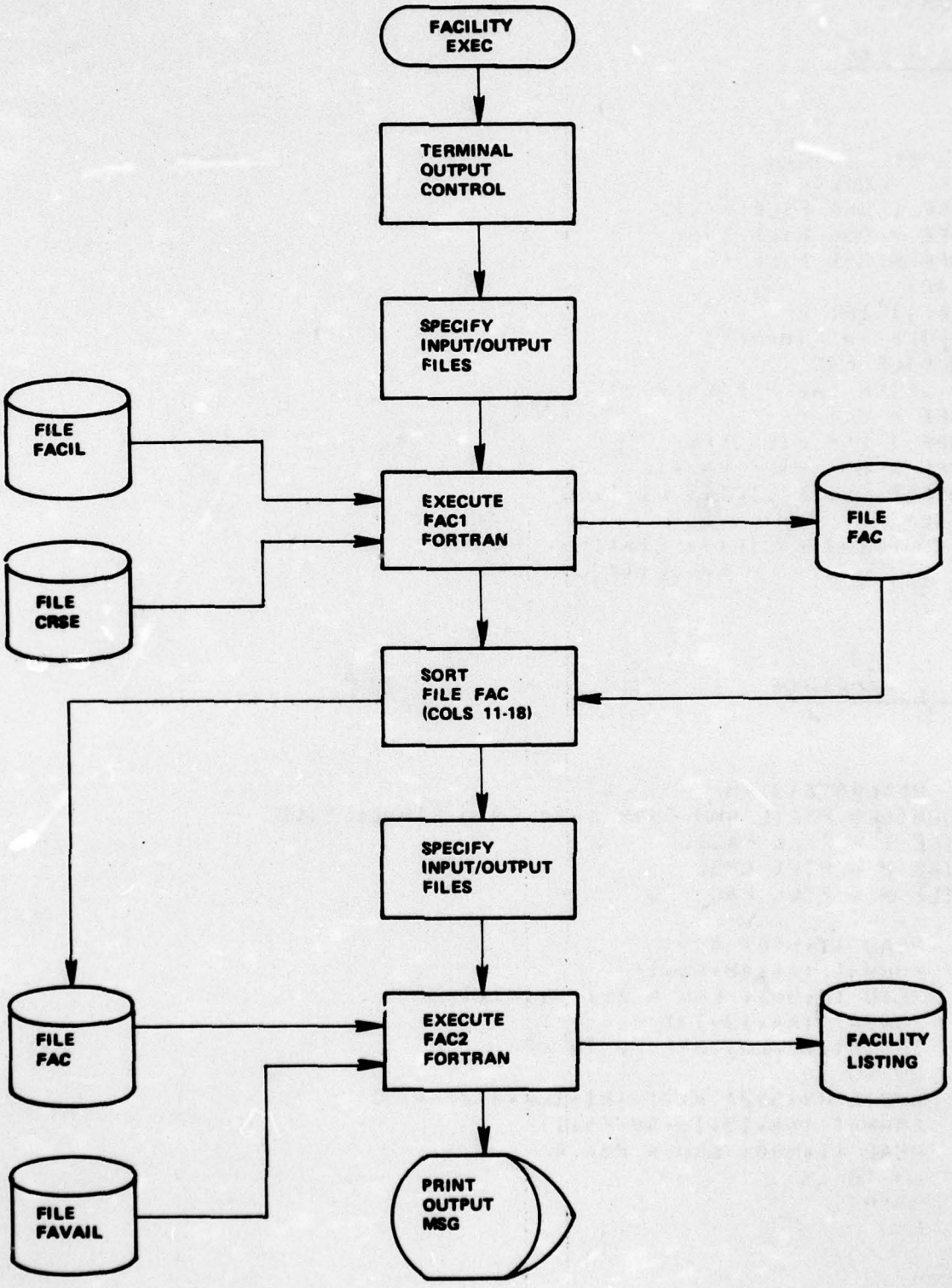


FIGURE VII-8

FACILITY PROGRAM LISTINGS

FACILITY EXEC

```
&TYPE OFF
FILEDEF * CLEAR
FILEDEF 1 DSK FILE FACIL
FILEDEF 2 DSK FILE CPSE
FILEDEF 4 DSK FILE FAC
RUN FAC1
&STACK 11 18
SORT FILE FAC (BRIEF)
ERASE FILE FAC
ALTER .FILE FAC P FILE * *
FILEDEF * CLEAR
FILEDEF 1 DSK FILE FAC
FILEDEF 2 DSK FILE FAVAIL
FILEDEF 4 DSK FACILITY LISTING
RUN FAC2
&PRI TO PRINT FACILITY LISTING, TYPE:
&PRI                P FACILITY LISTING
```

FAC1 FORTRAN

```
      REAL*8 A(3),B
C**COMBINES FACIL AND CPSE DATA INTO SINGLE FILE
C  FILE 1 = FILE FACIL
C  FILE 2 = FILE CPSE
C  FILE 4 = FILE FAC
C
      READ (1,500) A
  500 FORMAT (A4,A8,F5.1)
   10 READ (2,501, END = 25) B,IB1,IB2
  501 FORMAT (A4,I3,I3)
   15 IF (A(1) .EQ. B ) GO TO 20
      GO TO 10
   20 WRITE (4,502) A(1),IB1,IB2,A(2),A(3)
  502 FORMAT (A4,I3,I3,A8,F5.1)
   22 READ (1,500, END = 25) A
      GO TO 15
   25 STOP
      END
```

TABLE VII-14

FAC2 FORTRAN

```
      REAL*8 A(5),B(5),C
C
C**FORMATS FACIL LISTING DATA FOR OUTPUT
C  FILE 1 = FILE FAC
C  FILE 2 = FILE FAVAIL
C  FILE 8 = FACILITY LISTING
C
      WRITE (8,100)
 100  FORMAT ('  BLDG / RM   COURSE  CLASIZE  CONV  HRS/CONV',
     1'  TOT HRS  AVAIL HRS'/'  ---------   ------  -------  ',
     1'----  --------  -------  ----------')
      READ (1,99) A(1),IA1,IA,A(4),A(5)
 99   FORMAT (A4,I3,I3,A8,F5.1)
      A(3) = IA
      TA = A(3) * A(5)
      TOT = TA
 10   READ (1,99, END = 150) B(1),IB1,IB,B(4),B(5)
      B(3) = IB
      TB = B(3) * B(5)
      IF ( A(4) .NE. B(4) ) GO TO 50
      WRITE (8,101) A(4),A(1),IA1,IA,A(5),TA
 101  FORMAT (2X,A8,5X,A4,5X,I3,4X,I3,5X,F5.1,4X,F6.1)
 25   TOT = TOT + TB
      DO 30  I=1,5
 30   A(I) = B(I)
      IA1 = IB1
      IA = IB
      TA = TB
      READ (1,99, END = 175) B(1),IB1,IB,B(4),B(5)
      B(3) = IB
      TB = B(3) * B(5)
      IF ( A(4) .NE. B(4) ) GO TO 90
      WRITE (8,102) A(1),IA1,IA,A(5),TA
 102  FORMAT (15X,A4,5X,I3,4X,I3,5X,F5.1,4X,F6.1)
      GO TO 25
 50   READ (2,104) C,IC
 104  FORMAT (A8,I5)
      WRITE (8,105) A(4),A(1),IA1,IA,A(5),TA,IC
```

TABLE VII-14 (Continued)

FAC2 FORTRAN

```
105   FORMAT (2X,A8,5X,A4,5X,I3,4X,I3,5X,F5.1,4X,F6.1,4X,I5/)
55    DO 60  I=1,5
60    A(I) = B(I)
      IA1 = IB1
      IA = IB
      TA = TB
      TOT = TA
      GO TO 10
90    READ (2,104) C,IC
      WRITE (8,102) A(1),IA1,IA,A(5),TA
      WRITE (8,106) TOT,IC
106   FORMAT (48X,'------'/42X,'TOT = ',F6.1,4X,I5/)
      GO TO 55
150   READ (2,104) C,IC
      WRITE (8,105) A(4),A(1),IA1,IA,A(5),TA,IC
      CALL EXIT
175   READ (2,104) C,IC
      WRITE (8,102) A(1),IA1,IA,A(5),TA
      WRITE (8,106) TOT,IC
      CALL EXIT
      END
```

TABLE VII-14 (Continued)

MODAVAIL EXEC DESCRIPTION

The purpose of MODAVAIL EXEC is to modify the number of available hours associated with either an instructor or a facility in the LP input data file SCRR DATA. MODAVAIL uses the CSS EDIT command to effect the file change. The EDIT command copies the SCRR DATA file into core. Subsequent edit requests examine and modify the core image of the file and then write the file to disk.

The initial EXEC statement suppresses printing of all CSS command statements and all EXEC control statements. Next the output file for the FORTRAN program MODIFY is specified and MODIFY is executed. The EXEC file CHG which has been created by MODIFY is then executed. CHG EXEC uses the CSS EDIT command to modify the file SCRR DATA. First, the line containing the specified instructor ID or facility name is located in the file by the LOCATE request. Then the available hour figure contained in the file is replaced with a new figure using the EDIT command CHANGE request. An error in data entry will result in a NO CHANGE message, which indicates that SCRR DATA has not been changed. The final statement in the CHG EXEC writes the updated SCRR DATA file to a writable disk. CHG EXEC is erased following execution. If the user indicates that additional availability modifications are required, MODAVAIL EXEC branches to statement two and reexecutes MODIFY and CHG EXEC. If further changes are not required, a message informing the user how to rerun the SCRR model is printed.

MODIFY FORTRAN DESCRIPTION. The purpose of MODIFY FORTRAN is to create the CHG EXEC file containing the user-specified availability information.

The user specifies the name of the instructor of facility, current available hours, and the new available hours. This information is formatted within several CSS EDIT command request statements and written on output file CHG EXEC. The CHG EXEC file created by MODIFY is shown below.

```
CHG EXEC

&STACK KT
&BEGSTACK
L  /RHS/
L  /ID/
C  /ID/              CHRS/ID          NHRS/
FILE
&ENDSTACK
EDIT SCRR DATA
```

MODAVAIL EXEC FLOWCHART

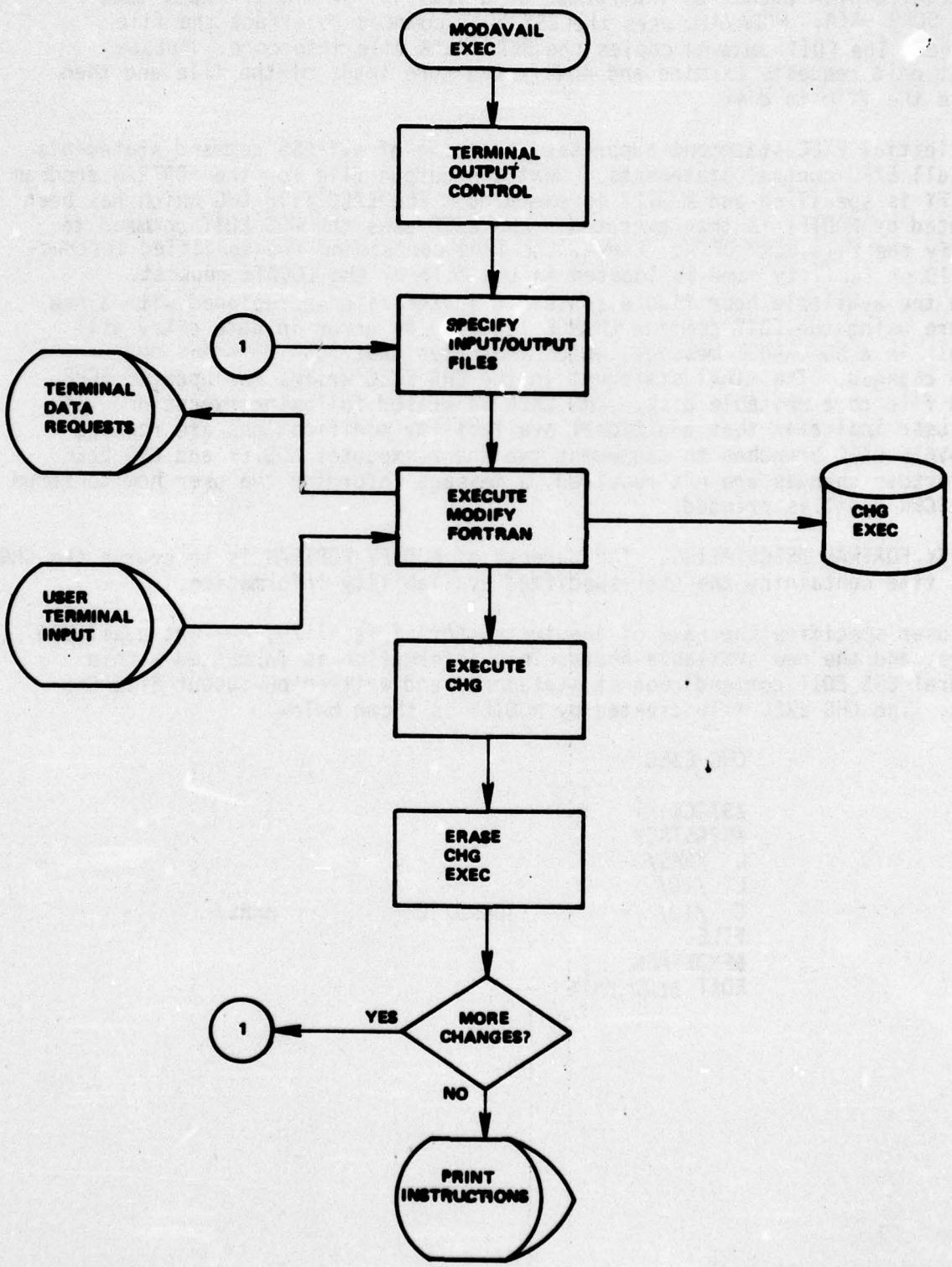


FIGURE VII-9

MODAVAIL PROGRAM LISTINGS

MODAVAIL EXEC

```
&TYPE OFF
-CHG FILEDEF 8 DSK CHG EXEC
FILEDEF 6 CONO
RUN MODIFY
EXEC CHG
CSS ERASE CHG EXEC
&PRI ANY MORE CHANGES? (YES/NO)
&READ ARGS
&IF &1 NE NO &GOTO -CHG
&PRI CHANGES TO AVAILABLE HOURS HAVE BEEN INCORPORATED.
&PRI TO RERUN SCRR MODEL  TYPE:  SCRR1
&EXIT
```

TABLE VII-15

MODIFY FOPTRAN

```
      REAL*8 RES,NME,ID
      WRITE(6,9)
 9    FORMAT ('ENTER NAME OF INSTRUCTOR OR FACILITY: ')
      READ (5,10) RES
 10   FORMAT (A8)
 5    WRITE (6,11)
 11   FORMAT ('ENTER CURRENT AVAILABLE HOURS, NEW AVAILABLE HOURS: ')
      READ (5,12) ICHR,INHR
 12   FORMAT (2I99)
      IF (ICHR .GT. 9999) GO TO 20
      IF (INHR .GT. 9999) GO TO 20
 25   READ(1,16,END=30) ID,NME
 16   FORMAT(A8,1X,A8,16X)
      IF ( RES .EQ. NME ) GO TO 60
      GOTO 25
 30   WRITE (8,13) RES,RES,ICHR,RES,INHR
 13   FORMAT ('&STACK KT'/'&BEGSTACK'/'L. /RHS/'/'L /',A8,'/'/'C /',A8,
     18X,I4,'/',A8,8X,I4,'/'/'FILE'/'&ENDSTACK'/'EDIT SCHR DATA')
 50   CALL EXIT
 60   WRITE(8,13) ID,ID,ICHR,ID,INHR
      GO TO 50
 20   WRITE (6,15)
 15   FORMAT ('AVAILABLE HOURS LIMITED TO FOUR (4) DIGIT NUMBER! TRY ',
     1'AGAIN.')
      GO TO 5
      END
```

TABLE VII-15 (Continued)

REQMT EXEC DESCRIPTION

The purpose of REQMT EXEC is to provide the model user with the capability to modify course utilization, class capacity, and course length expressed in contact hours to assess the impact of these changes on total instructor and facility requirements. The EXEC utilizes three RAMIS RUN EXECUTIVE procedures to retrieve data from two RAMIS data files COURSE and FACIL. The RUN EXECs extract current utilization, capacity and length data from the data files and then factor these data by the user-specified percentage changes prior to calculating and filing instructor and facility requirements. RUN EXECUTIVES are cataloged RAMIS procedures capable of accepting variable parameter input values.

The initial REQMT EXEC statement suppresses printing of all CSS command statements and all EXEC control statements. The user is then asked if program operating instructions should be printed. If the response is positive, several lines of text are printed prior to each data request. A negative response will cause all text statements to be bypassed. Values are then requested for the utilization percent change, capacity change, and the length change.

After the requested data have been entered, output files are specified, RAMIS is attached, and three RAMIS RUN EXECUTIVES are executed. REQMT uses all three parameters (utilization, capacity, and length) and generates the file FILE INST which contains instructor contact hour requirements by instructor by course. REQMT1 uses only the utilization and capacity parameters and generates the file FILE CRSE which contains the class size and number of convenings by course. REQMT2 uses the course length parameter and generates the file FACIL which contains the contact hours requirements by facility by course. After REQMT2 has been executed, RAMIS is detached and a message which indicates that the required output files have been generated is printed.

REQMT RUNEXEC DESCRIPTION. The purpose of the REQMT RUNEXEC is to calculate instructor requirements in hours by course and by instructor incorporating the course parameter deltas as specified by the user through the REQMT EXEC. The basic data required to determine instructor requirements are stored in the RAMIS course file COURSE.

REQMT RUNEXEC is based on the RAMIS procedures SCRR INST1 and SCRR INST2 described earlier. The primary difference between the procedures is that the RUNEXEC factors the data retrieved from the course file by the user-specified percentages. REQMT output is written on FILE INST.

REQMT RUNEXEC is a cataloged procedure stored in the RAMIS file RAMEXEC. Maintenance of the RAMEXEC file to change a procedure can be performed in the UPDATE mode or in the SCAN mode through the usual RAMIS facilities. Alternately the file REQMT RUNEXEC can be modified, and the procedure recataloged by using the RAMIS command RAMISIN REQMT RUNEXEC.

REQMT1 RUNEXEC DESCRIPTION. The purpose of the REQMT1 RUNEXEC is to record the convening frequency and average class size for each course incorporating the user-specified course parameter deltas. The required ata are stored in the RAMIS course file COURSE.

REQMT1 RUNEXEC is based on the RAMIS procedure SCRR CRSE described earlier. The primary difference between the two procedures is that the RUNEXEC factors the data retrieved from the course file by the user-specified percentages. REQMT1 output is written on FILE CRSE.

REQMT2 RUNEXEC DESCRIPTION. The purpose of the REQMT2 RUNEXEC is to determine facility requirements in hours by course and by facility incorporating the course length delta specified by the user. The basic data are stored in the RAMIS facility file FACIL.

REQMT2 RUNEXEC is based on the RAMIS procedure SCRR FACIL described earlier. The primary difference between the two procedures is that the RUN EXEC factors the data retrieved from the facility file by the user-specified percentages. REQMT2 output is written on FILE FACIL.

REQMT EXEC FLOWCHART

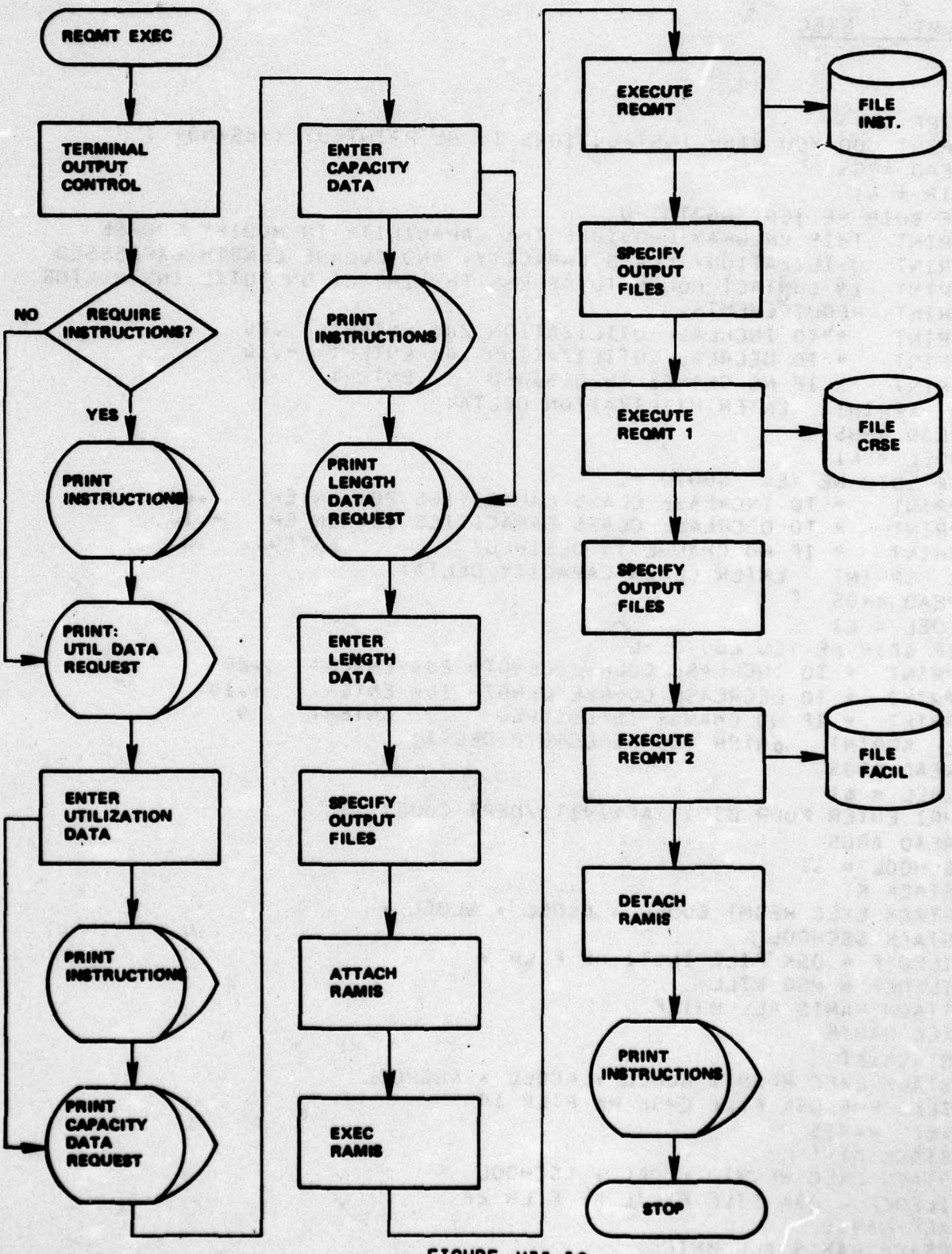


FIGURE VII-10

REQMT PROGRAM LISTINGS

REQMT EXEC

```
&TYPE OFF
&PRINT  DO YOU WISH INSTRUCTIONS TO BE PRINTED? (YES/NO) :
&READ ARGS
&DIR = &1
&IF &DIR NE YES  &GOTO -U
&PRINT  THIS PROGRAM PROVIDES THE CAPABILITY TO MODIFY COURSE
&PRINT  UTILIZATION, CLASS CAPACITY, AND COURSE LENGTH EXPRESSED
&PRINT  IN CONTACT HOURS TO ASSESS THE IMPACT ON TOTAL INSTRUCTOR
&PRINT  REQUIREMENTS.
&PRINT   * TO INCREASE UTILIZATION 20% ENTER:   .20
&PRINT   * TO DECREASE UTILIZATION 10% ENTER:  -.10
&PRINT   * IF NO CHANGE IS DESIRED     ENTER:    0
-U  &PRINT   ENTER UTILIZATION DELTA:
&READ ARGS
&UDEL = &1
&IF &DIR NE YES  &GOTO -C
&PRINT   * TO INCREASE CLASS CAPACITIES 20% ENTER:   .20
&PRINT   * TO DECREASE CLASS CAPACITIES 10% ENTER:  -.10
&PRINT   * IF NO CHANGE IS DESIRED          ENTER:    0
-C  &PRINT   ENTER CLASS CAPACITY DELTA:
&READ ARGS
&CDEL = &1
&IF &DIR NE YES &GOTO -L
&PRINT  * TO INCREASE COURSE LENGTH 20% ENTER:   .20
&PRINT  * TO DECREASE COURSE LENGTH 10% ENTER:   -.10
&PRINT  * IF NO CHANGE IS DESIRED       ENTER:    0
-L  &PRINT   ENTER COURSE LENGTH DELTA:
&READ ARGS
&LDEL = &1
&PRI ENTER FOUR DIGIT ACTIVITY/DEPT CODE:
&READ ARGS
&SCHOOL = &1
&STACK KT
&STACK EXEC REQMT &UDEL , &CDEL , &LDEL ,
&STACK &SCHOOL
FILEDEF 4 DSK FILE INST1 RE F LR 9
FILEDEF 6 MSG KILL
ATTACH RAMIS ALL BRIEF
EXEC RAMIS
&STACK KT
&STACK EXEC REQMT1 &UDEL , &CDEL , &SCHOOL
FILEDEF 4 DSK FILE CHSE RE F LR 10
EXEC  RAMIS
&STACK KT
&STACK EXEC REQMT2 &LDEL , &SCHOOL
FILEDEF 4 DSK FILE FACIL RE F LR 22
EXEC RAMIS
DETACH RAMIS ALL BRIEF
```

TABLE VII-16

REQMT EXEC

```
ERASE MSG KILL
FILEDEF 6 CONO
FILEDEF 1 DSK FILE INST2
FILEDEF 2 DSK FILE INST1
FILEDEF 9 DSK FILE INST RE F LR 34
RUN SCRRINST
&PRI REQUESTED CHANGES HAVE BEEN INCORPORATED.
&PRI TO RERUN SCRR MODEL  TYPE:  SCRR2
```

REQMT1 RUNEXEC

```
DELETE
FILE RAMEXEC
1=OUT=REQMT1,$
END
CATALOG REQMT1
TABLEF
FILE COURSE
PRINT CAP1 AND UCAP AND CONV1 AND HOLD
BY CDP
IF SCH IS &3
END
DEFINE
FILE HOLD
UCAP1/F4.2 = MIN(1.0,(.01*UCAP)+(UCAP*.01*(&1)))
CLASIZE/I3 = (CAP1 * UCAP1 * ((&2)+1.0)) + .5
END
TABLE
FILE HOLD
PRINT CLASIZE AND CONV1 AND SAVE BY CDP
END
FIN
END CATALOG
```

TABLE VII-16 (Continued)

REQMT RUNEXEC

```
DELETE
FILE RAMEXEC
1=OUT=REQMT.S
END
CATALOG REQMT
TABLE
FILE COURSE
PRINT CAP1 AND UCAP AND F1 AND F2 AND F3 AND HOLD BY CDP
IF SCH IS &4
END
DEFINE
FILE HOLD
IF F1 GT 150 GOTO INST
UCAP1/F4.2 = MIN(1.0,(UCAP*.01)+(UCAP*.01*(&1)))
AVGCLAS / I3 = (CAP1 * UCAP1 * ((&2)+1)) + .5
NOINST /I3 = MAX (1.0, (AVGCLAS / F1 ) + .5)
TOTHRS /I4 = NOINST *F2*((&3)+1);
FLAG /I1 = 1;
GOTO END;
INST;
ASN /I1 = F3
FLAG /I1 = 0
END
TABLE
FILE HOLD
STORE TOTHRS BY CDP
IF FLAG IS 1
RUN
FILE HOLD
SUM ASN AND HOLD BY CDP IF FLAG IS 0
END
DEFINE
FILE HOLD
INSTHRS /F5.1 = TOTHRS / ASN
END
TABLE
FILE HOLD
PRINT INSTHRS AND SAVE BY CDP
END
FIN
END CATALOG
```

TABLE VII-16 (Continued)

REQMT2 RUNEXEC

```
DELETE
FILE RAMEXEC
1=OUT=REQMT2,$
END
CATALOG REQMT2
TABLE
FILE FACIL
PRINT REQHRS AND HOLD BY CDP BY BLDG BY RM
IF SCH IS &2
END
DEFINE
FILE HOLD
RHRS/F6.1 = REQHRS*(REQHRS*(&1))
END
TABLE
FILE HOLD
PRINT RHRS AND SAVE BY CDP BY BLDG BY RM
END
FIN
END CATALOG
```

TABLE VII-16 (Continued)

SCRR2 EXEC DESCRIPTION

The purpose of the SCRR2 EXEC is to incorporate the resource requirement changes requested through the REQMT EXEC into the LP input data matrix and then rerun the SCRR model.

REQMT EXEC regenerates three RAMIS interface files - FILE INST, FILE FACIL, and FILE CRSE. To incorporate the data from these files into the LP data matrix, it is necessary to execute those executives which require any of these files as input. Therefore, the following executives are serially executed by the SCRR2 EXEC:

BLANK
IF2
SCRRSORT
COMB
SCRR1

SCRR2 PROGRAM LISTING

SCRR2 EXEC

```
&PRI THIS EXECUTIVE WILL INCORPORATE THE CHANGES
&PRI REQUESTED BY THE REQMT EXECUTIVE INTO THE LP
&PRI INPUT AND RERUN THE SCRR MODEL
&TYPE OFF
EXEC BLANK
EXEC IF2
EXEC SCRRSORT
EXEC COMB
EXEC SCRR1
```

TABLE VII-17

FIXCONV EXEC DESCRIPTION

The purpose of the FIXCONV EXEC is to modify the LP input data MATRIX1 DATA to fix the number of course convenings at the current level.

The MATRIX1 DATA file contains one BOUND data card image for each course in the current problem formulation. Columns 2-3 of this card image contain the bound type indicator. The indicator LO indicates that the number of convenings specified in columns 25-36 is a lower bound. This is equivalent to adding a constraint to the LP problem formulation specifying that the number of convenings must be greater than or equal to the number specified on the BOUND data card. FIXCONV EXEC uses the NCSS EDIT command to change the BOUND type indicator LO on all BOUND data cards in MATRIX1 DATA to the type indicator FX. FX indicates that the number of convenings specified on the BOUND data card is a fixed value.

FIXCONV PROGRAM LISTING

FIXCONV EXEC

```
&BEGSTACK
KT
C /LO/FX/ * *
FILE
&ENDSTACK
EDIT MATRIX1 DATA SB
&PRI COURSE CONVENINGS FIXED AT THE CURRENT LEVEL.
&PRI TO RERUN SCRR MODEL TYPE : SCRR1
&EXIT
```

TABLE VII-18

MAXCONV EXEC DESCRIPTION

The purpose of the MAXCONV EXEC is to modify the LP input data MATRIX1 DATA to remove the upper bound on course convenings previously established through the execution of FIXCONV EXEC.

MAXCONV EXEC reverses the data change made by FIXCONV EXEC. MAXCONV uses the NCSS EDIT command to change the BOUND type indicator FX on all BOUND data cards on MATRIX1 DATA to the type indicator LO.

If the type indicator is not FX when MAXCONV is executed, the program will respond with the message NO CHANGE.

MAXCONV PROGRAM LISTING

MAXCONV EXEC

```
&BEGSTACK
KT
C /FX SCH/LO SCH/ * *
FILE
&ENDSTACK
EDIT SCRH DATA SB
&PRI DATA MODIFIED TO OPTIMIZE COURSE CONVENINGS.
&PRI TO RERUN SCRH MODEL, TYPE:   SCRH1
```

TABLE VII-19

SCRR3 EXEC DESCRIPTION

The purpose of the SCRR3 EXEC is to construct the LP input data matrix SCRR DATA and then rerun the SCRR model software after data has been retrieved from the three RAMIS data files COURSE, INST, and FACIL by executing the executive RAMSCRR.

The following executives are serially executed by the SCRR3 EXEC:

BLANK
IF1
IF2
SCRRSORT
IF3
IF4
COMB
SCRR1

SCRR3 PROGRAM LISTING

SCRR3 EXEC

```
&IF &CORE GE 320 &SKIP 3
&PRI REQUIRED CORE IS 320K. PLEASE TYPE: SET CORE 320
&PRI THEN REISSUE COMMAND: SCRR
&EXIT
EXEC BLANK
EXEC IF1
EXEC IF2
EXEC SCRRSORT
&STACK KT
EXEC IF3
EXEC IF4
EXEC COMB
EXEC SCRR1
```

ICOMP EXEC DESCRIPTION

The purpose of ICOMP EXEC is to compute instructor requirements according to the technique outlined in CNTECHTRAINST 5311.1A CH-1.

ICOMP EXEC first requests the user to specify whether requirements are to be generated for a single course or for all courses within a devision/department. If the user selects the division option, computed instructor requirements are printed for all courses within the division. If the user selects the single course option, instructor requirements can be computed for a demand other than the current year demand. The system will print instructor requirements for the demand specified by the user.

ICOMP RUNEXEC DESCRIPTION

ICOMP RUNEXEC first extracts all class capacity, course convening, student/instructor ratio, and contact hour data from the RAMIS file COURSE. If a demand other than the current demand has been specified, the number of convenings is calculated by dividing the demand by the class capacity. ICOMP RUNEXEC then computes the number of instructors required at each student/instructor ratio. Fractional instructors greater than 0.2 are rounded to the next whole number. Total contact hour requirements are then determined by summing the product of the number of instructors required and the contact hours at each student/instructor ratio.

The total contact hours required are then divided by the convening frequency (number of convenings divided by 50) times 25 to determine total situations.

$$\frac{\text{TOT HR X \#CONV}}{50 \text{ X } 25} = \frac{\text{TOT HR}}{\text{CONV FREQ X } 25} = \text{TOT SITUATIONS}$$

Basic requirements are derived by multiplying the total situations by 1.10. Dividing basic requirments by 0.88 establishes total instructor requirements.

$$\frac{\text{TOT SITUATIONS X } 1.10}{0.88} = \text{TOTAL INST REQUIREMENTS}$$

ICOMP PROGRAM LISTINGS

ICOMP EXEC

```
&TYPE OFF
&PRINT TO CALCULATE INSTRUCTOR REQUIREMENTS FOR:
&PRINT *** A SINGLE COURSE  TYPE: CDP
&PRINT *** ALL COURSES WITHIN A DIVISION OR DEPT  TYPE: SCH
&READ ARGS
&FLD = &1
&IF &FLD EQ SCH &GOTO -SCH
&PRINT ENTER COURSE CDP NUMBER:
&READ ARGS
&VAL = &1
&GOTO -DMD
-SCH &PRINT ENTER DIVISION/DEPT CODE:
&READ ARGS
&VAL = &1
&DMD = 0
-DMD &IF &FLD EQ SCH &GOTO -RUN
&PRINT TO CHANGE THE ANNUAL DEMAND FOR COURSE &VAL FROM THE CURRENT
&PRINT LEVEL AS STORED IN THE DATA BASE TO A NEW VALUE,
&PRINT TYPE NEW DEMAND VALUE: ELSE TYPE 0.
&READ ARGS
&DMD = &1
-RUN
&STACK KT
&STACK EXEC ICOMP &FLD . &VAL . &DMD
FILEDEF 6 MSG KILL
ATTACH RAMIS
EXEC RAMIS
&STACK KT
&STACK L. /COMP/
&STACK F. 0
&STACK DE. 2
&STACK U. /COMP/
&STACK P 100
&STACK Q
EDIT MSG KILL
DETACH RAMIS
```

TABLE VII-21

ICOMP RUNEXEC

```
DELETE
FILE RAMEXEC
1=OUT=ICOMP.$
END
CATALOG ICOMP
TABLE
FILE COURSE
PRINT CAP1 AND CONV1 AND F1 AND F2 AND HOLD BY CDP
IF &1 IS &2
IF F1 IS-LESS-THAN 150
END
DEFINE
FILE HOLD
&&IF &3 EQ 0 GOTO CALC;
CONV/I3=&3/CAP1
&&CALC
NOINST/I3=MAX(1.0,(CAP1/F1)+.799)
TOTHRS/I4=NOINST*F2
END
TABLE
FILE HOLD
SUM TOTHRS AND HOLD BY CDP BY CONV1
&&IF &3 EQ 0 GOTO SK1;
BY CONV
&&SK1
END
DEFINE
FILE HOLD
TOTSIT1=(TOTHRS*CONV1)/1250
TOTREQ1/F6.2=(TOTSIT1*1.1)/.88
&&IF &3 EQ 0 GOTO SK2;
TOTSIT=(TOTHRS*CONV)/1250
TOTREQ/F6.2=(TOTSIT*1.1)/.88
&&SK2
END
TABLE
HEADING
COMPUTED INSTRUCTOR REQUIREMENTS                &DATE
FILE HOLD
PRINT TOTREQ1 AS INSTR.REQD.CUR.OMD
&&IF &3 EQ 0 GOTO SK3;
AND TOTREQ AS INSTR.REQD.&3.OMD
&&SK3
BY CDP
ON CDP SKIP-LINE
END
FIN
END CATALOG
```

TABLE VII-21 (Continued)

APPENDIX A

ALPHABETIZED PROGRAM LISTING INDEX

PROGRAM NAME		PAGE
SCRR	CRSE	VII-16
COURSE	DESC	V-12
FACIL	DESC	V-4
INST	DESC	V-7
ASSIGN	EXEC	VII-56
BLANK	EXEC	VII-22
CENTER	EXEC	VI-37
COMB	EXEC	VII-37
COURSE	EXEC	V-208
CRSRUN	EXEC	VI-33
CTRRUN	EXEC	VI-37
DBPROMPT		V-29
DBTAPERD		V-68
DISP	EXEC	VI-37
E1	EXEC	V-208
E2	EXEC	V-209
E3	EXEC	V-209
E4	EXEC	V-209
E5	EXEC	V-209
E6	EXEC	V-210
E7	EXEC	V-210
E8	EXEC	V-210
E9	EXEC	V-210
FACIL	EXEC	V-211
FACILITY	EXEC	VII-62
FCOURSE	EXEC	V-21
FINST	EXEC	V-211
FIXCONV	EXEC	VII-77
GPLISTF	EXEC	V-212
GP1	EXEC	V-212
ICOMP	EXEC	VII-81
IF1	EXEC	VII-27
IF2	EXEC	VII-30
IF3	EXEC	VII-34
IF4	EXEC	VII-36
INST	EXEC	V-213
MAXCONV	EXEC	VII-78
MODAVAIL	EXEC	VII-67
M13EXB	EXEC	V-110
M15EXB	EXEC	V-119
M15EXC	EXEC	V-120
M16EXB	EXEC	V-127
M17EXB	EXEC	V-135
M18EXB	EXEC	V-141

PROGRAM NAME		PAGE
M19EXB	EXEC	V-149
M20EXB	EXEC	V-155
M99EXA	EXEC	V-111
NDUMP	EXEC	V-213
PCB	EXEC	V-214
PCOURSE	EXEC	V-214
PFACIL	EXEC	V-214
PFCB	EXEC	V-215
PICB	EXEC	V-215
PINST	EXEC	V-215
PROCESS	EXEC	V-216
RAM	EXEC	V-216
RAMSCRR	EXEC	VII-16
RDUMP	EXEC	V-216
REFORM	EXEC	V-217
RELEASE	EXEC	V-217
REQMT	EXEC	VII-72
RUNPROC	EXEC	V-218
SCRR	EXEC	VII-11
SCRRSORT	EXEC	VII-32
SCRR1	EXEC	VII-44
SCRR2	EXEC	VII-76
SCRR3	EXEC	VII-79
SETSCH	EXEC	VII-7
SS	EXEC	VII-6
SUPER	EXEC	V-219
SUPERDUM	EXEC	V-219
SUPERX	EXEC	V-220
SUPRDUMP	EXEC	V-220
TPF	EXEC	VI-31
TPFCOPY	EXEC	VI-7
TPFMOD	EXEC	VI-19
TPFSORT	EXEC	VI-8
SCRR	FACIL	VII-17
SCRR	FAVAIL	VII-17
ASSIGN1	FORTRAN	VII-57
ASSIGN2	FORTRAN	VII-57
CONVRT	FORTRAN	VI-26
DECODE	FORTRAN	VI-22
DISPLAY	FORTRAN	VI-38
FAC1	FORTRAN	VII-62
FAC2	FORTRAN	VII-63
IF1	FORTRAN	VII-27
IF2	FORTRAN	VII-30
IF3	FORTRAN	VII-34
IF4	FORTRAN	VII-36
KEND	FORTRAN	VI-47
MODIFY	FORTRAN	VII-68
MSG	FORTRAN	VI-27
NCONT	FORTRAN	VII-50
SCRRFORM	FORTRAN	VII-46

PROGRAM NAME		PAGE
SCRRINST	FORTRAN	VII-19
SETSCH	FORTRAN	VII-8
SHOW	FORTRAN	VI-28
TPFMOD	FORTRAN	VI-20
SCRR	IAVAIL	VII-17
SCRR	INST1	VII-18
SCRR	INST2	VII-19
FCOURSE	PLI	V-171
FINST	PLI	V-174
MPGM3	PLI	V-177
PARAM	PLI	VI-32
PCOURSE	PLI	V-180
PFACIL	PLI	V-185
PGM80	PLI	VI-10
PGM93A	PLI	VII-51
PGM94A	PLI	VII-22
PGM94B	PLI	VII-23
PINST	PLI	V-189
SUPRPGM	PLI	V-194
COURSE1	RAM	V-221
COURSE2	RAM	V-221
COURSE3	RAM	V-221
FACIL1	RAM	V-222
FACIL2	RAM	V-222
INST1	RAM	V-222
INST2	RAM	V-222
RE48	RAM	VI-8
RE48X	RAM	VI-8
SCRR	RUN	VII-45
ICOMP	RUNEXEC	VII-82
REQMT	RUNEXEC	VII-74
REQMT1	RUNEXEC	VII-73
REQMT2	RENEXEC	VII-75
INDEX	RUNPROC	V-223
M13RP	RUNPROC	V-112
M15RPA	RUNPROC	V-121
M15RPB	RUNPROC	V-122
M15RPC	RUNPROC	V-123
M16RPA	RUNPROC	V-129
M16RPB	RUNPROC	V-130
M16RPC	RUNPROC	V-131
M17RP	RUNPROC	V-137
M18RPA	RUNPROC	V-144
M18RPB	RUNPROC	V-145
M18RPC	RUNPROC	V-146
M19RP	RUNPROC	V-151
M20RPA	RUNPROC	V-156
M20RPB	RUNPROC	V-157
M20RPC	RUNPROC	V-158